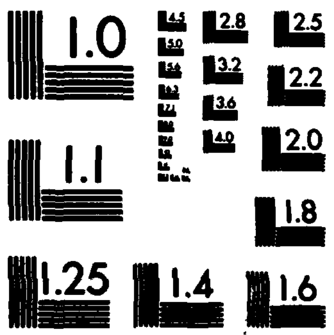
AD-A136 638 RULE-BASED STATISTICAL CALCULATIONS ON A DATABASE ABSTRACT(U) STANFORD UNIV CA DEPT OF COMPUTER SCIENCE N C ROWE JUN 83 STAN-CS-83-975 N00039-82-C-0250 1/2

UNCLASSIFIED F/G 5/2 NL


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

PHOTOGRAPH THIS SHEET

A136638

DTIC ACCESSION NUMBER

II

LEVEL

INVENTORY

Rept. No. STAN-CS-83-975

DOCUMENT IDENTIFICATION

Jun. '83

Contract N00039-82-C-0250

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR		
NTIS	GRA&I	☒
DTIC	TAB	☐
UNANNOUNCED		☐
JUSTIFICATION		
BY		
DISTRIBUTION /		
AVAILABILITY CODES		
DIST	AVAIL AND/OR SPECIAL	
A/1		

DISTRIBUTION STAMP

DTIC
S ELECTE D
JAN 9 1984
D

DATE ACCESSIONED

83 12 22 036

DATE RECEIVED IN DTIC

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDA-2

DTIC FORM OCT 79 70A

DOCUMENT PROCESSING SHEET

June 1983 Report No. STAN-CS-83-975

Rule-based Statistical Calculations on a Database Abstract

by

Neil Charles Rowe

Contract N00039-82-C-0250

Department of Computer Science

Stanford University
Stanford, CA 94305

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION None	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY None	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE None	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) STAN-CS-83-975	5. MONITORING ORGANIZATION REPORT NUMBER(S) Same
6a. NAME OF PERFORMING ORGANIZATION Stanford University 6b. OFFICE SYMBOL *(If applicable)* NA	7a. NAME OF MONITORING ORGANIZATION Same
6c. ADDRESS *(City, State and ZIP Code)* Department of Computer Science Stanford University Stanford, CA 94305	7b. ADDRESS *(City, State and ZIP Code)* Same
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Unknown 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Contract N00039-82-~~8~~C-0250
8c. ADDRESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE *(Include Security Classification)* Rule-based Statistical Calculations on a Database Abstract	
12. PERSONAL AUTHOR(S) Neil Charles Rowe	
13a. TYPE OF REPORT Dissertation 13b. TIME COVERED FROM ______ TO ______	14. DATE OF REPORT *(Yr., Mo., Day)* June 1983 15. PAGE COUNT 151
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*	
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL [illegible]	22b. TELEPHONE NUMBER *(Include Area Code)* 69-26412 22c. OFFICE SYMBOL ECEX 623

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Rule-based Statistical Calculations on a Database Abstract

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Neil Charles Rowe

June 1983

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

(Gio C. M. Wiederhold)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

(Bruce G. Buchanan)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

(S. Jerrold Kaplan)

Approved for the University Committee
on Graduate Studies:

(Dean of Graduate Studies & Research)

Abstract

The size of data sets subjected to statistical analysis is increasing as computer technology develops. Quick estimates of statistics rather than exact values are becoming increasingly important to analysts. We propose a new technique for estimating statistics on a database, a "top-down" alternative to the "bottom-up" method of sampling. This approach precomputes a set of general-purpose statistics on the database, a "database abstract", and then uses a large set of inference rules to make bounded estimates of other, arbitrary statistics requested by users. The inference rules form a new example of an artificial-intelligence "expert system". There are several important advantages of this approach over sampling methods, as is demonstrated in part by detailed experimental comparisons for two quite different databases.

Acknowledgments

This thesis is the result of many influences. In the summer of 1980 I was doing reading in physical clustering of database records, and it occurred to me that some sort of auxiliary structure containing statistics on all the records in one cluster might be useful. I put this idea aside until spring of 1982, whereupon Jerry Kaplan suggested I take another look at it and try to combine it with the notion of fast approximate answers to queries to a database. Since I had just been involved in a project involving probabilistic modelling, statistical queries seemed the obvious choice for studying appropriate approximate answers. Familiarity with the work of the Heuristic Programming Project at Stanford made the choice of a production-system architecture clear. Theoretical works in contract bridge impressed upon me the importance of absolute bounds on statistical inference in closed worlds.

Gio Wiederhold and Bruce Buchanan have been helpful in suggesting tough questions to address. Other people who have helped in one way or another include Jim Davidson, Kyu-Young Whang, Bob Blum, Harry Wong, John McCarthy (the one of Lawrence Berkeley Laboratory), Dorothy Denning, Persi Diaconis, Peter Cheeseman, and Jack Alpert.

Parts of this thesis have appeared in modified forms elsewhere. Chapter 1 is substantially [Rowe 83a], which is turn a revision of [Rowe 81]. Section 3.4.5 is extracted and revised from [Rowe 82], and much of chapter 6 has been submitted to the Second LBL Workshop on Statistical Database Management, 1983. Thanks to all referees for suggestions regarding these papers.

This work is part of the Knowledge Base Management Systems Project, under contract #N00039-82-G-0250 from the Defense Advanced Research Projects Agency of the United States Department of Defense. The views and conclusions contained in this document are those of the author and should not be interpreted as representative of the official policies of DARPA or the US Government.

"Publish?" Dan hunched forward, protecting the notebook with his knobby wrist. "No, I don't publish. It's not the point. It's not what I'm working for, my name in some AI journal, I don't have time, see?"

"But that's how you buy the time, publishing. How do you think somebody like Czernski got the Norbert Wiener Chair of Cybernetics at -- "

"Anyway, why should I? Roderick's mine, think I want to stick him in some AI journal for everybody to rip-off? He's private, he's not another toy for some toy company, I don't want to see him crammed inside some plastic Snoopy doll. I don't want him grabbed up by some Pentagon asshole to make smart tanks."

"Don't know what the hell you're talking about," Ben lit a cigarette. "Applications, what the hell do you care about applications? Feel like I'm sitting here with Alexander Graham Bell, he's invented this swell gadget only he's afraid to tell anybody about it, in case some loony uses it to make dirty phone calls."

John Sladek, Roderick, *Pocket Books, 1980*

Captain Kook: What in the world was THAT?!

Mr. Spook: That was a 7.895 Amplitude Shock Wave!

Captain Kook: That big? Are you sure?

Mr. Spook: Positive! Look at this instrument! See? The little hand fell off Donald Duck!

Mad Magazine, *1967 (in* The Pocket Mad, *Warner, 1974)*

Table of Contents

List of Figures

Chapter 1
Overview

But the Demon of the Second Kind continued to operate at a speed of three hundred million facts per second, . . . so he greedily read everything that flew out from under the diamond nib, the drinking songs of Quaidacabondish and the sizes of bedroom slippers available on the continent of Cob, with pompons and without, and the number of hairs growing on each brass knuckle of the skew-beezered flummox, and the average width of the fontanel in indigenous stepinfants, and the litanies of the M'hot-t'-ma-hon'h conjurers to rouse the reverend Blotto Ben-Blear, and the inaugural catcalls of the Duke of Zilch, and six ways to cook cream of wheat, and a good poison for uncles with goatees, and twelve types of forensic tickling, and the names of all the citizens of Foofaraw Junction beginning with the letter M, and results of a poll of opinions on the taste of beer mixed with mushroom syrup....

Stanislaw Lem, Cyberiada, *1967, trans. M. Kandel (Avon, 1976)*

1.1. Introduction

The critical aspects of statistical analysis of data are often the storage and access, not the statistical analysis routines themselves [Bates et al 82, Shoshani 82]. For very large data sets this suggests special attention to characteristics of secondary storage devices. Forthcoming devices such as videodisks, bubble memories, and special processors may improve time and space efficiency in the years ahead, but it will be difficult for such improvements to surpass those in processing power due to mere integration and scaling down of components in VLSI designs. It seems likely that secondary storage access will continue to be the bottleneck in statistical analysis of computer data[1].

Perhaps, however, we can trade off processing speed for storage. Statistical databases often have much redundancy in attribute values and statistics that can be predicted by other attribute

[1] And a corollary of Parkinson's Law seems to operate in statistical analysis: data expand to fill available memory.

values and statistics. If we can formulate this redundancy computationally, we may be able to put it into cheap processors and programs instead of expensive secondary storage. We may be able to create a much smaller database, a "database abstract" [Rowe 81], that preserves most of the information content of the original data. We can then use a number of "reasonable guess" rules from statistics (together with some "special case" checks) to infer (impute) statistical characteristics of the original data from the abstract -- that is, by an artificial-intelligence "expert system".

This approach shares some ground with "exploratory data analysis" (EDA) [Tukey 77]. EDA is a loosely related set of techniques for "getting a feel for data" in the initial stages of statistical analysis, emphasizing quick and rough estimates and visual displays. It is directed towards hypothesis generation, not hypothesis testing (or "confirmatory data analysis"), though it does not preclude subsequent use of the latter on the data if interesting phenomena are found worthy of further study. Our objective stated in the last paragraph is very close to this. But EDA is also relevant to this work in another way: we can support EDA activities by creating small, partial database abstracts for data analysts to explore rather than full databases. The advantages are speed of query answering and storage savings (perhaps allowing employment of a small personal computer).

There are, however, two disadvantages with our approach. First, our answers to statistical questions will usually be approximate rather that exact, just as with random sampling. But for the user doing EDA, estimates (perhaps in the form of rough graphs) are often perfectly satisfactory. Second, we require a "closed world" database with which to work; that is, a database that includes all possible data items of a given type, not just a sample of them. For instance, to reason about the population of the United States we need records for every person in the United States, and not just records for California or a 1-in-1000 random sample. However, with increasing computerization of routine data in all areas such a coverage can be quite possible.

1.2. Overview of the approach

Figure 1-1 shows our approach. We start with a user and a database. The database is preprocessed to create a "database abstract", a collection of simple statistics (mean, maximum, mode frequency, etc.) on important and frequently asked-about sets in the database. The user talks through an interface to the database abstract, asking it the same statistical questions he would ask the full database if he had more time (or space). If an answer is not in the database abstract, an estimate and bounds on that estimate are inferred for it from rules. These rules (400 of them in our current implementation) form an artificial-intelligence production system [Davis and King 76], and represent distinct pieces of domain-independent knowledge about statistical estimation from a variety of of very different sources. Some (e.g., EDA rules) are justified on intuitive criteria, but most can be derived mathematically, by theorem-proving methods, non-linear optimization, and maximum entropy theory. Rules can also be suggested by analogies to other rules.

The next chapter gives a demonstration of the implementation. Walker [Walker 80] has also studied database abstracts, in addressing how the abstraction operation interacts with the standard database query operations [Ullman 81] of selection, projection and join. But his work differs from ours in two fundamental ways: (a) it ignores statistical aggregates, and (b) it addresses the conditions for exact answers, not what you can say about an inexact answer.

1.3. What's wrong with sampling

Our approach provides a new alternative to random sampling for exploring a large data population at low cost. There are several serious disadvantages to attempting to estimate statistics on a population by statistics on a random sample of that population[2]:

1. The data may already be partially aggregated in means, counts, etc., as when there are large amounts of data from instrument readings in laboratory experiments. Sampling then is tricky, and may not be possible without detailed information about the preaggregation.

2. Sampling is inefficient in paging. Suppose we randomly sample m items from a database of p pages with k items per page on the average. We can model the apportionment of sample items to pages as a Poisson process where the expected number of pages examined in sampling is $p(1-e^{-m/p})$, approximately $p(1-(1-m/p)) = m$ for $m < p$. This will generally be considerably larger than m/k, the sampling ratio. So a sample of one thousandth of a database with a hundred items to a page will access one tenth the pages, not one thousandth.

   Because of this, Morgenstein [Morgenstein 80] has proposed randomized page assignment for databases. But this faces many problems. Randomization is very complex when joins are involved. Randomization is difficult to maintain on updates without reorganizing the whole database. Most important of all, randomization of database pages is will markedly degrade performance for nonstatistical queries on the same database, and thus is inappropriate for most large databases, which are multi-purpose.

[2]Most of these disadvantages apply to to estimation from *any* sample, random or not. For instance, stratified sampling designs can only possibly answer objections 3 and 6, and only partially.

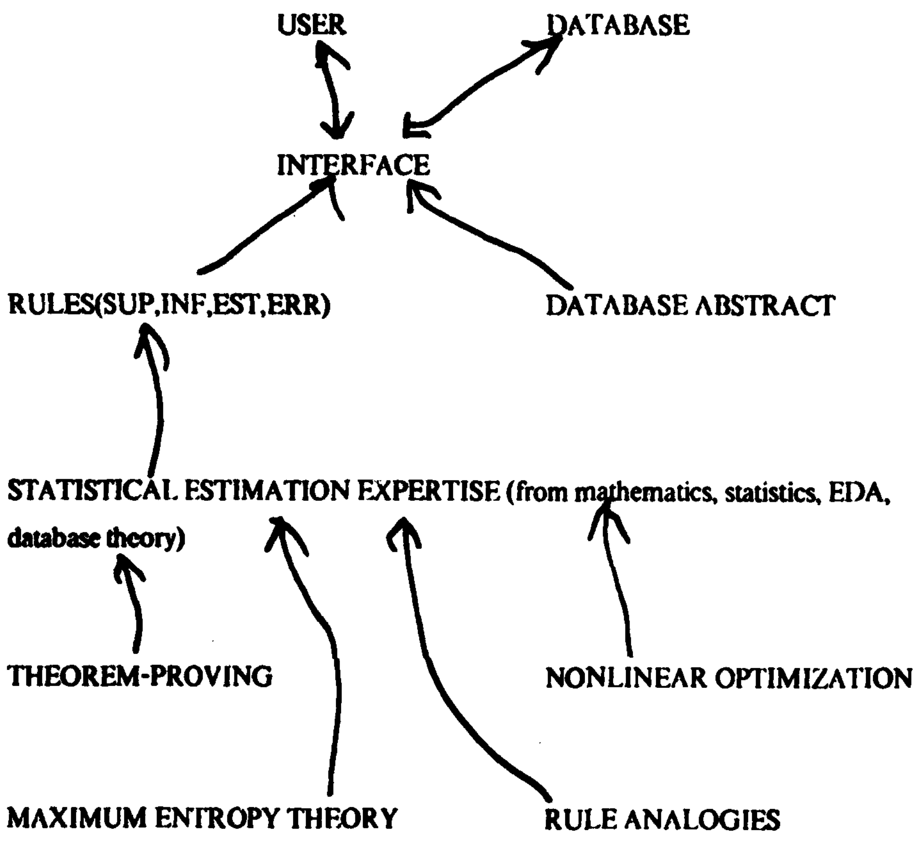


Figure 1-1: Block diagram of the system

3. Random sampling is also inefficient with indexes when they are used. Since one has no assurance that indexes list items in a statistically random order, usually one must assemble pointers to all the items in the set to choose randomly among them. This may require much temporary storage space for the pointers, and many index page accesses, depending on how the index is stored.

4. Sampling is a poor way to estimate extremum statistics like maximum, mode frequency, and bounds on distributional fits. Many applications can exploit such statistics.

5. Similarly, sampling is very poor for obtaining absolute bounds on statistics, which are important for many computer algorithms based on those statistics (cf. section 7.2.2.3).

6. Sampling is "brittle": given a sample, it is hard to speculate about properties of a subset, superset, or sibling of that set. This is serious because an EDA user may not be sure what he wants to look at, or want to explore clusters of related sets in the course of data analysis, or may simply make mistakes.

   A random sample of a set is unlikely to be a random sample of a superset or sibling. (We may be able to *use* it in a stratified random sample design for the superset or sibling, but such designs are highly data-specific.) So if we choose a set too restrictive in our initial query to a database, we must sample all over again for a superset, with all the paging inefficiences doubled. On the other hand, we must also resample if we choose too large a set to start with, for otherwise we're sampling a sample, a poor statistical design. (Erroneous generalizations are suggested from the accidental correlations of a sample of a sample.) Also, the removal of points not in the subpopulation from the sample may leave too few points for sufficiently reliable inferences.

7. A random sample doesn't have semantics. That is, it is of interest only as a random sample and not as an entity in its own right as a set created by set intersections might be.

8. Sampling in the artificial world of a database makes less sense than in natural, real world populations. Databases are "closed worlds" containing finite amounts of well-characterized data, where each datum is (in principle at least) equally accessible. And with increasing computerization of routine clerical records in many different parts of society, computer database are coming more and more to be complete in covering all of a real-world population. On the other hand, real world populations, as for example populations of people, have fluid boundaries and members of differing accessibilities. The idea of knowing the mean exactly for some set only makes sense in a (nonsampling) database -- we can only be more or less sure of the mean in the real world.

Our approach of a database abstract of precomputed statistics plus some inference rules compares favorably to random sampling in regard to each of the abovementioned points:

1. The database abstract *is* aggregated data.

2. Once set up, the database need not be paged at all. Paging of the abstract will be low (see section 1.7.2). And setup can be quite efficient -- each page can be fetched in turn, and every item on a page examined.

3. Database index pages are used efficiently for the same reasons.

4. Inference rules can handle extremum statistics (e.g. maximum) nearly as well as normative statistics (e.g. mean).

5. Inference rules explicitly infer bounds.

6. Inference rules gracefully handle extensions of a set to supersets and restrictions of a set to subsets; many rules explicitly address these cases.

7. Sets in the database abstract have an explicit semantics.

8. Inference explicitly takes into account the closed world nature of databases.

1.4. About rules

We now discuss the inference rules used with the database abstract.

1.4.1. The querying language

We use a set-descriptive language for queries, in the style of relational algebra (as opposed to predicate calculus). Figure 1-2 gives a formal specification. Queries S(C,F,R) consist of a statistical aggregate operator S applied to three arguments: a database relation R, a class C (or set) of items within that relation, and an attribute F (or field) of those items. (The statistics S are defined in Appendix A.) Rules are substitutions for a query of a particular form by a mathematical function of the results of other queries.

This query language is equivalent to most relational-algebra languages for database queries [Ullman 81], with a few additional statistical operators; see Appendix B for details.

1.4.2. Describing answers

Rules may give exact or, usually, inexact answers for queries. When inexact for a statistic S, we describe the probability distribution with four items of information (and always four items):

An arbitrary single-statistic query is in the form S(C,F,R) where

- S is a single-attribute aggregate statistical operator
- C is a set of tuples (a "class"), or "rows" of a relation
- F is an attribute or field (which may be virtual), or "column" of a relation
- R is a relation

S, C, R, and F are specified as follows:

- S :: SIZE | MAX | MIN | MEAN | SIGMA | MEDIAN | MODE | SIZEUNIQUE | MODEFREQ | MODEFREQ2 | LEASTFREQ | MAXGAP | MINGAP | MAXEVENDEV | MINEVENDEV | KEYS

  (see Appendix A for definitions of these statistics)

- C :: first-order-class | NOT(C) | AND(C,C) | OR(C,C)

  (where NOT corresponds to set complement, AND to set intersection, and OR to set union)

- F :: schema-fieldname | ARITHOP(F) | ARITHOP2(F,F) | ABSTRACTION(F) | VECTORIZE(F,F)
- R :: relation-name | JOIN(R,R,F)
- ARITHOP :: SQUARE | SQRT | LOG | ANTILOG | RECIPROCAL | ABS
- ARITHOP2 :: PLUS | DIFFERENCE | TIMES | QUOTIENT | MAX | MIN
- ABSTRACTION :: domain-dependent nonnumeric function on an attribute

Figure 1-2: The query language

- An upper bound on S (alias SUP-S)
- A lower bound on S (alias INF-S)
- An estimate of S (alias EST-S)
- The expected standard error of that estimate (alias ERR-S)

The bounds are absolutely guaranteed. The estimate is guaranteed (see 1.8.2) within some criterion (say 10%) on some finite number of queries chosen by the system designer.

1.4.3. The rule taxonomy

Our four hundred rules can be categorized along four different dimensions:

- the statistic dimension (whether it is mean, maximum, standard deviation, mode frequency, etc.)
- the characteristic dimension (whether it is an exact answer, a bound, an estimate, or a standard error of an estimate)
- the computational dimension (what form of queries it applies to) -- see Figure 1-3
- the derivation dimension (where we got it) -- see Figure 1-4

As an example, consider the rule that the largest item in the intersection of two sets cannot be any larger that the minima of the maxima of the two sets for some numeric attribute. On the statistic dimension, this is a rule for a maximum statistic; on the characteristic dimension, this is a SUP (upper bound); on the computational dimension, this is a set intersection rule of the tuple-class-decomposition type; and on the derivation dimension, this is a theorem derivable from basic mathematics. Or consider the rule that the mode of a set can be estimated as the mode of its largest subset. This is a mode rule on the statistic dimension; an EST (reasonable guess) on the characteristic dimension; an upwards inheritance rule (see 3.4.5) on the computational dimension; and a maximum entropy rule on the derivation dimension. Appendix C gives rule examples for all the items in the taxonomies of figures 1-3 and 1-4.

1. intraquadruple rules
2. statistic-statistic rules
3. row (item class) decomposition
   a. subset inheritance
   b. set intersection
   c. set union
   d. set complement
4. column (attribute expression) decomposition
   a. unary 1-to-1 operators
   b. other unary operators
   c. binary operators on corresponding values
   d. vectorization of corresponding values
   e. operations with constants
5. relation (join) decomposition
6. unusual inheritances
   a. upwards inheritance
   b. lateral inheritance
   c. diagonal inheritance
   d. attribute-hierarchy inheritance
   e. rule inheritance
7. canonical query rearrangement
8. the closed-world rule

Figure 1-3: The computational dimension of rules

1. basic mathematics
2. probability and statistics
    a. definitions
    b. theorems
    c. extrema of definitions and theorems
        i. bounds on values
        ii. independence assumptions
        iii. linearity assumptions
        iv. nonlinear optimization
        v. entropy maximization
3. database theory
    a. functional dependencies
    b. theory of inference compromise
        i. small samples and trackers
        ii. exploiting uniqueness
        iii. Diophantine (integer) equations
4. new rules from old
    a. rule composition
    b. rule rearrangement
    c. theorem proving
    d. analogics
5. reasoning from prototypical examples

Figure 1-4: The derivation dimension of rules

1.4.4. A production system architecture

The production-system architecture frequently used in artificial intelligence expert systems [Davis and King 76] is strongly suggested here:

- The derivation and computational dimensions of the taxonomy show much variety.
- Rules represent highly modular pieces of knowledge.
- So much heterogeny enables synergistic effects where several very different rules together lead to surprising results that could not be foreseen by examining any of the rules independently.
- There is no "complete" set of rules. There are always special cases formulable in a more powerful additional rule, perhaps automatically (see 1.6.4); additional rules can improve performance for particularly common or important queries. And in moving to a smaller computer we may want to remove rules that aren't sufficiently cost-effective.
- Production rules let us make a conceptually clean break between database-independent knowledge (the rules) and database-dependent knowledge (the database abstract).

Note the productions are invoked by "backwards chaining"; to establish the value for a statistic you need to establish values for others first.

1.5. Two processing examples

To make things clearer, we show two examples of a number of different rules contributing to answering a query.

1.5.1. Example 1: a set size on the ships database

First, suppose a query asks for the number of American tankers in a database of ship information[3]. Assume that basic statistics on American ships and tankers separately are available, but none for American tankers. With this information, we cannot uniquely determine the size of the set intersection. But for a user who is satisfied with an estimate, we can try the following lines of reasoning:

1. An intersection of two sets can't be any larger than the smaller of the two. For 1000 American ships and 5000 tankers, there cannot be more than 1000 American tankers (a SUP).

[3]The statistic values here have been made up to illustrate our points. Actually, according to the definitive source of [Lloyd's 82], there were in 1982 8645 tankers, 6133 American ships, and 361 American tankers. Other statistical information about these ships has not been published.

2. Set sizes are nonnegative, so there are no fewer than zero (an INF).

3. For an estimate, use a simple log-linear model. If there are 20,000 ships in the database, expect 5000 * 1000 / 20000 = 250 American tankers (an EST).

4. For the standard error (ERR) of this estimate, find the standard deviation of a truncated exponential distribution (the maximum-entropy distribution) consistent with SUP, INF, and EST. The math is complicated and we won't go into it here.

5. If the mode tanker nationality occurs 140 times, there cannot be more than 140 American tankers (a SUP). Analogously, find the mode frequency of the ship type field for the set of American ships.

6. Similarly, if the least frequent ("antimode") tanker nationality occurs 5 times, there cannot be fewer than 5 American tankers (an INF).

7. Occasionally we may know a superset containing a set intersection. For instance, all American tankers may have identification codes of a certain type, hence the number of ships with those type codes is an upper bound on the number of American tankers (a SUP).

8. We can infer bounds and estimates from "range analysis" of arbitrary numeric attributes. Suppose the length range of tankers is from 300 to 1000 feet and that of American ships 50 to 400 feet. Then American tankers must have lengths 300 to 400 feet.

   If we have statistics on length subdivisions of ships, we can upper-bound the size of this set (a SUP). Statistics for any range that includes 300-to-400, like 260-to-420, will do too (we may partition on deciles, etc.). We can lower this upper bound if we know the maximum distance between successive American tankers in this range, or the maximum deviation of values across the range from some standard distribution. For instance, we may know that values are never off more than 10% of range from where they would be in a perfectly even distribution, in which case for our 260-to-420 example range, the 300-to-400 range can contain no more than 100/160 + .1 + .1 = 82.5% of the points in the 260-to-420 range.

9. We can set up Diophantine (integer-solution) equations for mean computations and find sets of distinct points (as opposed to ranges) consistent with those equations. Suppose there is a numeric code for each of four basic ship types (freighters, tankers, bulk carriers, and miscellaneous) -- say 2, 3, 8, and 11 respectively. Suppose the mean of this code field for American ships is 2.3. Then we can solve simultaneously the two equations:

   $$2n_1 + 3n_2 + 8n_3 + 11n_4 = 2.3 * 1000$$
   $$n_1 + n_2 + n_3 + n_4 = 1000$$

   where variable n_2 represents the number of American tankers which we are looking for. For these equations there can only be a number of American tankers which is a multiple of 3 up to 300, excluding 297. Hence 300 is a SUP.

   In general, there are many ways to include extra information leading to fewer solutions. We discuss this inference method in detail in [Rowe 83b].

To give an answer to the original query, we must combine the results of these rules. To do this we make the cumulative SUP the minimum of all the SUPs found, and we make the cumulative INF the maximum of all the INFs found[4]. In this case, assuming items 7 and 8 do not apply, the cumulative SUP is 140, and the cumulative INF is 5. Since there is only one EST (rule 3) and one ERR (rule 4), we take these to be the cumulative EST and cumulative ERR; section 1.6.1 explains what if there are more than one.

Once obtained, this query answer is useful for many other related queries. The size of the union of two sets is the size of the first set plus the size of the second set minus the size of their intersection. The size of the intersection of three sets can be obtained from statistics on the intersection of two of those sets first, together with statistics on the third.

1.5.2. Example 2: a mean on the medical database

As a second example, we give a query on our medical (rheumatological) database, asking the mean of the serum cholesterol, per visit, for all male patients that were taking more than 10 units of prednisone. We assume we know statistics on the set of all patient visits, the set of male patient visits, and the set of visits by patients *not* taking significant amounts of prednisone (i.e., less than 10).

1. To get an estimate (EST) of the mean of the intersection of two sets, we weight the means of the two sets by the reciprocals of the sizes of the two sets. (Justification is that the smaller set affects more the nature of the intersection.) Assume that we know there are 550 male patient visits with a mean of 350 for cholesterol per visit, and 130 patients taking insignificant prednisone with a mean of 240. We need statistics of patients taking significant prednisone:

    a. For 1000 total patient visits, 1000-130 = 870 had significant prednisone.

    b. For a cholesterol mean of 310 over all patient visits, the mean for significant prednisone visits must be the weighted difference

    $$(310*1000 - 240*130) / (1000-130) = 320$$

    Hence our weighted average for the intersection is

    $$[(350/550) + (320/870)] / [(1/550) + (1/870)] = 338$$

2. We would also like bounds (SUP and INF) on this estimate. First we must know bounds on the size of the intersection set in order to know *how much* it can resemble its two parents.

[4]If $x \leq a$ and $x \leq b$, then $x \leq \min(a,b)$; if $x \geq a$ and $x \geq b$, then $x \geq \max(a,b)$.

a. If there are 550 male visits and 870 significant-prednisone visits, there can be no more than 550 of the intersection.

b. But since there are 1000 visits total, the intersection can only "subtract out" 1000-870 = 130 visits maximum from the 550 male visits, hence a lower bound on the intersection size is 550-130=420.

c. We can get a better lower bound from knowing the mode (greatest) frequency of prednisone values for the set of male visits. If it is 13, and only six distinct prednisone dosages for this patient population exist that can be considered negligible, then a lower bound on the intersection size is 550-(13*6) = 472.

d. Given the mean of prednisone dosages for the set of male visits and the number of such visits, we can again obtain two linear Diophantine equations to solve.

Hence 472 to 550 items are in the intersection set.

3. We now find bounds on the mean of the intersection. For an upper bound, we pretend that 78 items were removed from the male visit set, all equal to the minimum cholesterol level among low-prednisone dosages, let us suppose to be 138:

$$[(550*350) - (78*138)] / (550-78) = 385$$

Similarly, if the maximum cholesterol level among low-prednisone visits is 740 the lower bound on the mean is

$$[(550*350) - (78*740)] / (550-78) = 286$$

4. We can narrow these bounds if we know some distribution fits. That is, if we know the tolerance with which the distribution of cholesterol values for male patient visits fits an even distribution, and it is tight, we can bound the mean of any fraction k/550 of the points of this distribution, $0 \leq k \leq 78$, and use this number rather than the minimum (138) and maximum (740) used in the above calculation. (We don't have space for the math here.)

5. A very different way to get bounds is if we have marked in advance as "unusual" particular statistics on intersections of two sets. If we define "unusual" as deviating more than 10% in estimate from the actual value, then any statistic with no marking must be estimatable within 10%, and this narrows its bounds. In this example, no marking would mean an upper bound of 338 + 34 = 372, and a lower bound of 338 - 34 = 304.

As the above examples may suggest, a variety of sources of knowledge can contribute to a query answer. The mathematics for the rules is not complex, but has rarely been formalized before. Note it is difficult to predict which particular reasoning path will be critical in obtaining bounds on the answer, and thus many rules need to be tried on the same query even though only a few will matter in the end.

1.6. More about rules

1.6.1. Conflict resolution

As with other production systems [Nilsson 80], we must specify action when more than one rule applies to the same query. Bounds rules are easy -- we just apply all of them separately, and intersect the answer ranges. With estimates and standard errors we do two things in our implementation. First, we try as much as possible to establish priority schemes among rules so that there are "weak" rules that are defaults, used only when no "strong" rules apply. (And some "strong" rules are in turn defaults of some "super-strong" rules.) Second, in the few instances where more than one EST or ERR rule still applies to the same query, we execute the rules separately, model the results as truncated normal distributions, assume statistical independence, and use the standard formulae, which in the two-rule case are:

$$\sigma^2 = 1/[(1/\sigma^2_1) + (1/\sigma^2_2)]$$
$$x = \sigma^2 (x_1/\sigma_1^2 + x_2/\sigma_2^2)$$

1.6.2. Subqueries and caching

Queries may invoke many subqueries. To avoid infinite loops, we check subqueries against a stack of active queries, terminating analysis if a match is found. For efficiency in addition, we check new queries and subqueries against cache of previous queries and their answers. Cached answers may be saved over a session if users tend to concentrate on particular sets, kept at the end of a session for the next one, or pooled among a user group with similar interests. Items cached frequently enough could be added to the database abstract permanently.

1.6.3. Rule compilation, lower level

"Levels of knowledge" occur frequently in artificial-intelligence expert systems [Hart 82]. Upper levels represent general but hard-to-use knowledge that can be compiled in a computationally expensive operation into a more efficient form. For our system there is both a level below the rules to which they can be compiled into an efficiently executable form, discussed below, and a level above of general knowledge from which rules can be created, discussed in the next section.

As suggested by part 1.5, rules can work on different subqueries in parallel. Each can be assigned a processor, and subquery answers pooled in a common cache. In addition, rule condition testing can be made more efficient by a decision tree [McDermott, Newell, and Moore 78]. Usually large classes of rules can be eliminated by inspection; for instance, a query involving only the intersection of sets with respect to a database attribute doesn't need rules on set unions or complements, or virtual attributes.

1.6.4. Rule compilation: upper level

Another kind of rule compilation is the creating of rules from underlying theories. There are four basic approaches: rearrangement of an existing rule or functional composition of existing rules, symbolic optimization via theorem-proving to get bounds rules, entropy maximization to get estimates and standard-error rules, and analogies to previous rules to get both. All can be automated in a symbolic algebra system to varying degrees.

Given the maximum, minimum, mean, and median of a set, what is the largest possible value of the standard deviation? That is a SUP question for our system, but it is also a quadratic optimization problem. There are standard solution techniques [Gill, Murray, and Wright 81] given exact values for certain statistics. But with only approximate values denoted by ranges we require a kind of symbolic optimization which is much trickier, more like theorem-proving, and whose difficulty varies markedly from case to case.

Rules for estimates and standard errors are another thing altogether. In information systems in general, the best guess for a parameter is that with least information content [Shore and Johnson 81]. Formalizing this leads to the calculus of variations and a general solution involving Lagrange multipliers (see Appendix to [Shore and Johnson 81]). For instance, if we know the maximum, minimum, mean, and standard deviation of a set, the maximum entropy distribution is of the form $ae^{b(x-c)^2}$, where the constants a, b, and c can be determined uniquely. From this concrete distribution we then calculate any statistic we want to estimate.

Rules can also be found by analogy. The most obvious examples are maximum and minimum rules, where additions are replaced with subtractions, maxima with minima, and minima with maxima in the text of the rule. Analogies have been used extensively to obtain rules for our implementation. [Lenat 82] gives many other sophisticated ideas for analogy generation. But analogies can be misleading, and rules postulated must always be rigorously checked.

1.6.5. Algebra on quadruples

Since we express query answers as quadruples, we need a special algebra for arithmetic operations on them. For bounds, the first half of the quadruples, we can use ideas from interval analysis [Nickel 69, Rall 81], a branch of numerical analysis. For estimates and standard errors, we can interpret the quadruples as a truncated normal distribution (the maximum-entropy assumption), and use the formulas of [Haugen 68]. (Some adjustments must be made to his formulas to model closed world effects, like the mean of the sum of corresponding values for two numeric fields being *exactly* the sum of the means of the two fields instead of just an approximation, because one is drawing without replacement from a population.)

1.6.6. Our methods in distributed systems

The accuracy of estimates by these methods may vary considerably. When an estimate is unsatisfactory, the user should be able to go to the full database to get the exact answer. (Thus is true of our current implementation.) Thus our methods have particular application to distributed computer systems. In particular, we envision the database abstract and rules at local nodes, perhaps within small personal computers, with the database at a remote site only accessed when occasionally.

1.6.7. Easy extensions

Nulls that represent unknown values can be treated by taking statistics on the nonnull portions of the set, and inferring upwards to characteristics of the full set including the nulls. Inexact data can be formulated as quadruples, and treated by our special algebra directly.

1.7. The database abstract as a database

A database abstract plus rules can provide significant savings in both space and time over use of a full database for statistical computation.

1.7.1. Storage space

Database abstract entries are best grouped by sets. A bit string header for each set can indicate which statistics are kept and to how many bits of accuracy, and the values can follow in a compact form. Sets can be accessed by an index on their names; any set with at least one stored statistic will be put in this index.

Rule storage should be comparatively negligible. Rules are short and simple, and use very few symbols which can be encoded in very few bits, using the taxonomy. Triggering conditions can be compacted in decision trees, with potential parallelisms indicated by a few additional pointers; since we anticipate 1000 or so rules in even sophisticated systems, pointers need not be large. The rule interpreter itself need not be large.

Though the database abstract is a compression of a database, other compression methods may apply too. We can store low order bits for a statistic value, and infer high order bits by inference rules. That is, use a statistic on American ships as a "base register" value, and keep only the "offset" to the American tankers value with the statistics for American tankers.

The number of database abstract entries necessary to achieve a certain level of answer accuracy is difficult to say, though we are developing a theory. But note information theory cannot be cheated: a database abstract can only contain a limited number of bits, for answering with limited accuracy a limited number of nonredundant queries.

1.7.2. Time considerations

Our system can be quite fast if desired. Cached information from previous queries makes a big difference. We expect page faults to be low for the database abstract because:

- it will be smaller (generally, much smaller) than the full database, and all or part of it might reside permanently in main memory;
- usually there are not many sets relevant to a query (just the ones explicitly mentioned in it), hence many fewer retrievals than for the same query on the full database;
- putting all the statistics for the same set on the same page greatly increases locality of references.

1.8. Loading the database abstract

Choosing which statistics on what sets to store in the database abstract involves art as well as science, but some guidelines are possible.

1.8.1. Chosing the first-order classes

Sets representing simple concepts, e.g. "tankers", "American ships", "ships in the Mediterranean", are what we call first-order sets. While their statistics can sometimes be close to statistics on entire database relations (important things to have a lot of information for), it is statistically unlikely this will be a good estimate in general. So we must keep many statistics about first-order sets explicitly in the database abstract. Then when there is sufficient extra room we can include second-order sets (the intersections and unions of two first-order sets) and higher-order sets. Our approach works best for databases where attributes are correlated only is simple ways and few such higher-order sets are needed to capture subtleties. Note if there are n first-order classes there are $O(n^2)$ intersections of any two of those, $O(n^3)$ intersections of any three of those, and so on -- numbers increase rapidly.

First-order sets may be dictated by the needs of a user community. When choice is possible, they should be large enough to matter (say 10 items or more), and should represent reasonably even subdivisions of a database, consistent as possible with the way human beings cluster concepts.

1.8.2. Closed world reasoning

We mentioned an important idea in the last point of section 1.5.2. A useful trick for setup of the database abstract is to only enter "unusual" statistics, defined to mean that the rule-inferred value is within, say, 10% of its actual value. Since this involves effort in advance, we only check this for a limited "guarantee" set of queries. Generally this means only queries on sets larger than a certain minimum and relatively simple in description (like all queries involving three or fewer sets). So we have a new and powerful inference rule for answering queries, the "Closed World Rule": if a query is in the guaranteed query set and the answer is not in the database abstract, then the answer found by inference rules is within 10% (or else the answer would have been loaded).

1.8.3. Monotonicity

A complication of the preceding rule is that when new information is placed in the database abstract it may lessen the accuracy of answers to previous queries, causing what we call nonmonotonicity. For example, suppose we estimate the mean of the intersection of three sets by the average of the means of the three sets. Suppose we add to the database abstract the mean of the intersection of two of those sets, where this mean is far off from the mean of three sets taken together. We will get a poorer answer for the mean of the three sets after adding this new information. An answer that was within 10% before and did not need to be represented explicitly in the database abstract may now need to be, and the closed-world rule cannot hold.

There is a way out if we can impose a consistent partial ordering on all queries, where query A is partially ordered with respect to to query B if B is a subquery generated in processing A. If we then load the database abstract in some linearization of the partial ordering we cannot interfere with the closed-world rule. This means loading first-order set statistics before second-order, second-order before third, and so on. It also means ordering different statistics on the same sets, as mean before maximum, and standard deviation before mean.

A consistent partial ordering does require restriction of the class of possible inference rules. We are trading off the power of a number of miscellaneous "backwards" inference rules for one really powerful rule, the Closed World rule. (We can still use the backwards rules in answering queries, just not during loading.)

1.9. Handling updates to the database

When the full database is updated, the database abstract must remain consistent. This is not a problem for much statistical data since much of it is never updated [Shoshani 82]. But we can handle it by a set of update rules; [Koenig and Paige 81] provides a formal framework for developing them. Some updates are much easier to handle than others (e.g. to mean, standard deviation, set size), only requiring knowledge of the value updated, whereas others (e.g. median and mode) usually require expensive recomputation on the full database and hence may not be practical.

1.10. Evaluation

Evaluation of the performance of our system is a complex issue. In chapter 6 we will show for two particular databases that our methods compare favorably to several simpler alternatives in regard to both space and time, without sacrificing large amounts of accuracy.

1.11. Extensions and applications

- We can add various kinds of correlation statistics and rules to estimate them.
- We can include reasoning about possible distinct values.
- We can use graphics or linguistic hedges to capture fuzziness of query answers.
- When sufficiently standardized, the database abstract and rules can be put into hardware.
- Our set-size rules can estimate selectivities for general query optimization (cf. [Christodoulakis 81]).
- Rules provide ideas for compromise methods for work in inference security of statistical databases [Rowe 83b]. But more importantly, our system provides a testbed for this research, much of which has focused on small numbers of inference rules in compromise, ignoring synergism between quite different sorts of rules.
- Rules raise important issues for artificial intelligence. They allow extension of the notion of inheritance to statistical properties, filling a key gap in representation-of-knowledge research, and demonstrating the adequacy of sets as the building block of a knowledge representation system [Rowe 82].

1.12. Organization of the thesis

Chapter 2 gives a demonstration of our implementation for the two test databases. Chapters 3 and 4 describe our methods: chapter 3 covers the rules, and chapter 4 the database abstract. Chapter 5 concerns our specific implementation of these methods, and chapter 6 presents a formal evaluation of their success. Finally, chapter 7 points out some additional extensions and applications.

Chapter 2
Demonstration

Dost see thy leg? I'll take thy leg away from thy stern if thou speakest to me of the merchant service again. Merchant service indeed!

Herman Melville, Moby Dick

We have a demonstration system under Interlisp on a DEC-20. Two test databases were used, one containing merchant shipping data, and one containing medical-patient clinic-visit data. As discussed, the system attempts to answer statistical queries about a database. It gives fast answers, at the expense of accuracy, in the form of quadruples. A good deal of reasoning goes on beneath the surface of this protocol, discussed in chapters 3, 4, and 5.

2.1. The merchant shipping database

The database used for development consists of a single relation containing properties of merchant ships. In what follows, INMED denotes the set of ships in the Mediterranean; NATLI, Liberian ships; and ALI, ships of a particular subclass of tankers. The database abstract first includes simple statistics on all first-order (single-word-name) sets. No closed-world guarantees are offered, and no correlations between attributes are exploited. The system does not actually understand English; we have paraphrased the queries to make it easier to see what is going on, printing this after the "=" sign.

```
    10-(SETQ ABSTRACT 'FIRST.ORDER)
```
= We want to use a database abstract consisting only of statistics on sets with single-word names.

```
    11-(SETQ GUARANTEE 'NOGUARANTEE)
```
= We do not want to use any 20% closed-world guarantee (see section 1.8.2).

[We now print out some statistics on the first-order sets we will be querying, to give a flavor of what facts the rule-based reasoning works with. We have edited out statistics on the latitude and ship-type attributes of these sets as they are not relevant to the demonstration.]

12-(STUDY 'ALI)
= List the statistics on the set of type ALI tankers.
(SIZE 121 MAXLONG 179 MINLONG 2 MEANLONG 43.5 SIGMALONG 36.7 MEDIANLONG 31 MAXGAPLONG 27 MINGAPLONG 1 MODELONG 24 MODEFREQLONG 19 SIZEUNIQUELONG 44 LEASTFREQLONG 1 MODEFREQ2LONG 18 MAXEVENDEVLONG .03 MINEVENDEVLONG -.56 MODENAT UR MODEFREQNAT 33 SIZEUNIQUENAT 27 LEASTFREQNAT 1 MODEFREQ2NAT 15)

[Read these codes as follows (see Appendix A for more explanation of the statistics):
SIZE 121 = there are 121 type-ALI ships
MAXLONG 179 = their maximum longitude is 179 degrees
MINLONG 2 = their minimum longitude is 2 degrees
MEANLONG 43.5 = their mean longitude is 43.5
SIGMALONG 36.7 = their standard deviation is 36.7
MEDIANLONG 31 = their median is 31
MAXGAPLONG 27 = the largest gap between successive ordered longitudes is 27
MINGAPLONG 1 = the smallest gap between successive ordered longitudes is 1
MODELONG 24 = the mode longitude is 24
MODEFREQLONG 19 = the frequency of the mode is 19
SIZEUNIQUELONG 44 = there are 44 distinct longitude values
LEASTFREQLONG 1 = the least frequent longitude occurs once
MODEFREQ2LONG 18 = the second most common longitude occurs 18 times
MAXEVENDEVLONG .03 = fits an even distribution with upper limit .03
MINEVENDEVLONG -.56 = fits an even distribution with lower limit -.56
MODENAT UR = most common nationality is UR (Russian)
MODEFREQNAT 33 = its frequency is 33
SIZEUNIQUENAT 27 = there are 27 distinct nationalities
LEASTFREQNAT 1 = the least common nationality occurs once
MODEFREQ2NAT 15 = the second most common nationality occurs 15 times]

13-(STUDY 'NATLI)
= List statistics on the set of Liberian ships.

(SIZE 90 MAXLONG 168 MINLONG 3 MEANLONG 46.3 SIGMALONG 29.0 MEDIANLONG 47 MAXGAPLONG 29 MINGAPLONG 1 MODELONG 49 MODEFREQLONG 21 SIZEUNIQUELONG 32 LEASTFREQLONG 1 MODEFREQ2LONG 10 MAXEVENDEVLONG .06 MINEVENDEVLONG -.55 MODENAT LI MODEFREQNAT 90 SIZEUNIQUENAT 1 LEASTFREQNAT 0 MODEFREQ2NAT 0)

[To get a statistical estimate in our system, one invokes the function ANS with three arguments: query set, queried statistic, and query attribute (the latter being null for SIZE and KEYS statistics). Our queries in this demonstration take up four lines:

1. the formal query definition in our query language, what is actually typed to our system

2. a paraphrase in English (after an "=" sign)

3. the answer quadruple returned by the system, using the database abstract and rules

4. the actual answer, obtained from going to the original data and computing from it]

14–(ANS '(AND ALI NATLI) 'SIZE)
= How many type ALI Liberian tankers are there?
(GUESS: 18.0 GUESS-ERROR: 3.855017 UPPER-LIMIT: 24 LOWER-LIMIT: 11)
(ACTUAL ANSWER IS 11)

15–(ANS '(OR ALI NATLI) 'SIZE)
= How many ships are either type ALI or Liberian?
(GUESS: 193.0 GUESS-ERROR: 3.855017 UPPER-LIMIT: 200 LOWER-LIMIT: 187)
(ACTUAL ANSWER IS 200)

16–(ANS '(AND ALI NATLI) 'MEAN 'LONG)
= What's the mean longitude of a type ALI Liberian tanker?
(GUESS: 44.88269 GUESS-ERROR: 33.04689 UPPER-LIMIT: 168 LOWER-LIMIT: 3)
(ACTUAL ANSWER IS 48.54545)

17–(ANS '(OR ALI NATLI) 'MEAN 'LONG)
= What's the mean longitude of ships that are either type ALI or Liberian?
(GUESS: 44.67642 GUESS-ERROR: 3.083323 UPPER-LIMIT: 50.23529 LOWER-LIMIT: 26.975)
(ACTUAL ANSWER IS 44.465)

[Notice from a comparison of query 14 with query 15, and a comparison of 16 with 17, that the unions give better estimation results than the intersections. This is true almost always for any two query sets.]

[Now let's introduce a third set, the set of all ships in the Mediterranean (on a particular date).]

18-(STUDY 'INMED)
= Give statistics for all ships in the Mediterranean.
(SIZE 366 MAXLONG 79 MINLONG 3 MEANLONG 34.1 SIGMALONG 14.3 MEDIANLONG 31 MAXGAPLONG 8 MINGAPLONG 1 MODELONG 24 MODEFREQLONG 83 SIZEUNIQUELONG 46 LEASTFREQLONG 1 MODEFREQ2LONG 64 MAXEVENDEVLONG .14 MINEVENDEVLONG -.29 MODENAT LI MODEFREQNAT 69 SIZEUNIQUENAT 38 LEASTFREQNAT 1 MODEFREQ2NAT 68)

[The following query answer was obtained without use of binary rearrangement rules discussed in section 5.6.4, which were implemented in our system after this protocol was run; we wish to demonstrate comparison to queries 30, 31, 35, and 36. Hence bounds are not any better than for query 14.]

19-(ANS '(AND INMED (AND ALI NATLI)) 'SIZE)
= How many Liberian type-ALI tankers are in the Mediterranean?
(GUESS: 14.39708 GUESS-ERROR:1.051749 UPPER-LIMIT: 24 LOWER-LIMIT:0)
(ACTUAL ANSWER IS 7)

20-(ANS '(AND INMED (AND ALI NATLI)) 'MEAN 'LONG)
= What's the mean longitude for Liberian ships in the Mediterranean of type ALI?
(GUESS: 39.51238 GUESS-ERROR: 25.39134 UPPER-LIMIT: 79 LOWER-LIMIT: 3)
(ACTUAL ANSWER IS 50.71429)

[Comparing the last two answers to those for queries 14 and 16, we see that accuracy of SIZE and MEAN decreases with the number of sets intersected. This is a useful rule of thumb for any statistic.]

[Besides intersections (ANDs) and unions (ORs), we can also reason about set complements.]

21-(ANS '(AND (NOT INMED) (AND ALI NATLI)) 'MEAN 'LONG)
= What's the mean longitude for Liberian ships of type ALI not in the Mediterranean?
(GUESS: 49.60453 GUESS-ERROR: 42.37534 UPPER-LIMIT: 176 LOWER-LIMIT: 6)

(ACTUAL ANSWER IS 44.75)

[And we can also reason about nonnumeric attributes like ship nationality. Here for example is an estimation of a mode frequency.]

22 – (ANS '(AND ALI INMED) 'MODEFREQ 'NAT)
= What's the mode frequency of the nationalities of ALI ships in the Mediterranean?
(GUESS: 20.49946 GUESS-ERROR: 14.43376 UPPER-LIMIT: 78 LOWER-LIMIT: 2)
(ACTUAL ANSWER IS 16)

[We can also reason about generalizations on the values of a nonnumeric attribute. For instance, we can categorize ship nationality as one of three world regions, and estimate statistics on these region values.]

23 – REGIONGEN
= What regions are nationality codes assigned to?
(OTHER (US LI PL PN GC IZ RP YO IR MY SN NA KU CU SA BR IN GB) NORTHEUROPE (NO UR RO UK PO BU CH FI DA GE SZ) MED (IT SP CY GR FR TU EG AG))

[Meaning US through GB are assigned to region "OTHER", NO through SZ to region "NORTHEUROPE", and IT through AG to region "MED".]

24 – (ANS '(AND ALI INMED) 'MODEFREQ '(REGION NAT))
= What's the mode frequency of the regions of the nationalities of ALI ships in the Mediterranean?
(GUESS: 46.5509 GUESS-ERROR: 9.317242 UPPER-LIMIT: 78 LOWER-LIMIT: 26)
(ACTUAL ANSWER IS 37)

[We can also ask questions about sets whose statistics we know nothing about, provided we know a superset they are contained within. For instance, we have defined in our system the set of ships in the Adriatic (INADRIATIC) to be a subset of ships in the Mediterranean (INMED).]

25 – (ANS '(AND ALI INADRIATIC) 'MEAN 'LONG)
= What's the mean longitude of ALI ships in the Adriatic?
(GUESS: 38.81484 GUESS-ERROR: 27.84864 UPPER-LIMIT: 179 LOWER-LIMIT: 3)
(ACTUAL ANSWER IS 21.93939)

[We can also query virtual attributes derived from arithmetic operations on actual numeric attributes.]

26 – (ANS 'ALI 'MEAN '(PLUS (SQUARE LONG) (SQUARE LAT)))
= What's the mean of the sum of the squares of the latitude and longitude for type ALI ships?
(OUR ANSWER IS 4485.661)
(ACTUAL ANSWER IS 4485.661)

55 – (ANS 'ALI 'MEAN '(SQRT (PLUS (SQUARE LONG) (SQUARE LAT))))
= What's the mean distance of ALI-type ships from 30N5W?
(GUESS: 51.0 GUESS-ERROR: 12.3 UPPER-LIMIT: 57.1 LOWER-LIMIT: 6.0)
(ACTUAL ANSWER IS 42.34673)

[We now include statistics on all intersections of two first-order sets in the database abstract, and ask (19) and (20) again.]

27 – (SETQ ABSTRACT 'SECONDORDER)
= We want a second-order-intersection abstract [that is, one consisting of statistics on the intersection of any two first-order sets as well as statistics on every first-order set].

29 – (SETQ CACHE NIL)
= Forget all the previous query answers [we might give better answers because of that cached information, which might make the subsequent results look better than they deserve to be].

30 – (ANS '(AND INMED (AND ALI NATLI)) 'SIZE)
= How many Liberian ships in the Mediterranean are type ALI?
(GUESS: 10.87381 GUESS-ERROR: .3065485 UPPER-LIMIT: 11 LOWER-LIMIT: 7)
(ACTUAL ANSWER IS 7)

31 – (ANS '(AND INMED (AND ALI NATLI)) 'MEAN 'LONG)
(GUESS: 41.34377 GUESS-ERROR: 18.90695 UPPER-LIMIT: 79 LOWER-LIMIT: 14)
= What's the mean longitude for Liberian ships in the Mediterranean of type ALI?
(ACTUAL ANSWER IS 50.71429)

[Answers are better for this "second-order" abstract than the "first-order" abstract, for the same queries, as may be observed by comparing 14 with 30, and 15 with 31.]

[We now restore the original database abstract, but add a 20% guarantee query set to include all intersections of two first-order sets.]

32 – (SETQ ABSTRACT 'FIRSTORDER)

= Use the first-order (single-word-name sets) abstract again.

33 – (SETQ GUARANTEE 'SECONDORDER)

= Now apply a 20% guarantee on all answers (see section 1.8.2).

34 – (SETQ CACHE NIL)

= Again, throw away all previous query answers so as to not help in getting subsequent results.

35 – (ANS '(AND INMED (AND ALI NATLI)) 'SIZE)

(GUESS: 5.733211 GUESS-ERROR: .9619619 UPPER-LIMIT: 11 LOWER-LIMIT: 0)

= How many Liberian ships in the Mediterranean are type ALI?

(ACTUAL ANSWER IS 7)

36 – (ANS '(AND INMED (AND ALI NATLI)) 'MEAN 'LONG)

= What's the mean longitude for Liberian ships in the Mediterranean of type ALI?

(GUESS: 39.51238 GUESS-ERROR: 11.32716 UPPER-LIMIT: 79 LOWER-LIMIT: 3)

(ACTUAL ANSWER IS 50.71429)

[Answers are not as good as with the second-order abstract (compare query 35 with 30, and 36 with 31), but better than the plain first-order abstract without guarantees (compare 35 with 14, and 36 with 15).]

2.2. The rheumatology-patient database

The data for rheumatology patients clinic visits was subjected to a detailed formal study of the performance of our system, discussed in chapter 6. We give here a few processing examples, in a more systematic format than in the previous section. We believe that set intersections are the most important kinds of composite queries to a database system, and thus performance of our system for statistics on them is important.

6 – *(SETQ ABSTRACT (QUOTE FIRST.ORDER))

= Use a first-order database abstract.

7 – *(SETQ GUARANTEE (QUOTE NOGUARANTEE))

= Do not use any 20% closed-world guarantee.

[The following nine queries represent the same statistics on a different set, the set of all male patient visits where low cholesterol was measured. Except for the first (set size) statistic, they are all refer to the prednisone dosage attribute, measured in integer units. MALE denotes the male patient visit records, and LOCHOL the low-cholesterol visits.]

8 – *(ANS '(AND MALE LOCHOL) SIZE)
= How many visits involved male patients with low cholesterol?
(OUR ANSWER IS 33)
(ACTUAL ANSWER IS 33)

9 – *(ANS '(AND MALE LOCHOL) 'MEAN 'PREDNISONE)
= What was the mean prednisone dosage for those visits?
(GUESS: 17.42672 GUESS-ERROR: 13.84462 UPPER-LIMIT: 80 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 17.57576)

10 – *(ANS '(AND MALE LOCHOL) 'SIGMA 'PREDNISONE)
= What was the standard deviation of the prednisone values?
(GUESS: 13.54828 GUESS-ERROR: 1.534365 UPPER-LIMIT: 24.68412 LOWER-LIMIT: 0)
(ACTUAL ANSWER IS 14.94237)

11 – *(ANS '(AND MALE LOCHOL) 'MAX 'PREDNISONE)
= What was the maximum prednisone dosage in those visits?
(GUESS: 77.85685 GUESS-ERROR: 2.143146 UPPER-LIMIT: 80 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 60)

12 – *(ANS '(AND MALE LOCHOL) 'MIN 'PREDNISONE)
= What was the minimum?
(GUESS: 3.143146 GUESS-ERROR: 2.143146 UPPER-LIMIT: 80.0 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 1)

13 – *(ANS '(AND MALE LOCHOL) 'SIZEUNIQUE 'PREDNISONE)
= How many distinct prednisone dosages were there?
(GUESS: 16.2106 GUESS-ERROR: 5.484827 UPPER-LIMIT: 20 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 15)

14 – *(ANS '(AND MALE LOCHOL) 'MODEFREQ 'PREDNISONE)
= What was the frequency of the most common dosage?

(GUESS: 10.08819 GUESS-ERROR: 8.948928 UPPER-LIMIT: 33 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 5)

15 – *(ANS '(AND MALE LOCHOL) 'MODEFREQ2 'PREDNISONE)
= What was the frequency of the second most common dosage?
(GUESS: 5.044097 GUESS-ERROR: 2.020726 UPPER-LIMIT: 11 LOWER-LIMIT: 0)
(ACTUAL ANSWER IS 4)

16 – *(ANS '(AND MALE LOCHOL) 'LEASTFREQ 'PREDNISONE)
= What was the frequency of the least most common dosage?
(GUESS: 3.0 GUESS-ERROR: 1.16743 UPPER-LIMIT: 11 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 1)

[The next nine queries represent different statistics on the same set, in this case the set of all male patient visits where the temperature was normal in addition to there being low measured cholesterol. Again, the attribute queried (except for the first query) is prednisone dosage. The code MEDTEMP denotes the set of all visits with normal temperature, and as before MALE represents the set of male patient visits and LOCHOL the set of low-cholesterol visits.]

18 – *(ANS '(AND (AND MALE LOCHOL) MEDTEMP) 'SIZE)
= How many visits involved male patients with low cholesterol and normal temperature?
(OUR ANSWER IS 33)
(ACTUAL ANSWER IS 17)

19 – *(ANS '(AND (AND MALE LOCHOL) MEDTEMP) 'MEAN 'PREDNISONE)
= What was the mean prednisone dosage for those visits?
(GUESS: 17.44304 GUESS-ERROR: 13.17575 UPPER-LIMIT: 120 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 17.23529)

20 – *(ANS '(AND (AND MALE LOCHOL) MEDTEMP) 'SIGMA 'PREDNISONE)
= What was the associated standard deviation?
(GUESS: 13.49634 GUESS-ERROR: 1.549613 UPPER-LIMIT: 48.36783 LOWER-LIMIT: 0)
(ACTUAL ANSWER IS 15.349)

21 – *(ANS '(AND (AND MALE LOCHOL) MEDTEMP) 'MAX 'PREDNISONE)
= What was the maximum dosage?
(GUESS: 61.57157 GUESS-ERROR: 0.0 UPPER-LIMIT: 120 LOWER-LIMIT: 1)

(ACTUAL ANSWER IS 60)

22 – *(ANS '(AND (AND MALE LOCHOL) MEDTEMP) 'MIN 'PREDNISONE)
= What was the minimum?
(GUESS: 3.143146 GUESS-ERROR: 0.0 UPPER-LIMIT: 80.0 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 1)

23 – *(ANS '(AND (AND MALE LOCHOL) MEDTEMP) 'SIZEUNIQUE 'PREDNISONE)
= How many distinct prednisone dosages were there?
(GUESS: 13.5 GUESS-ERROR: 7.216878 UPPER-LIMIT: 26 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 12)

24 – *(ANS '(AND (AND MALE LOCHOL) MEDTEMP) 'MODEFREQ 'PREDNISONE)
= What was the frequency of the most common dosage?
(GUESS: 11.20344 GUESS-ERROR: 8.660253 UPPER-LIMIT: 33 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 2)

25 – *(ANS '(AND (AND MALE LOCHOL) MEDTEMP) 'MODEFREQ2 'PREDNISONE)
= What was the frequency of the second most common dosage?
(GUESS: 5.601722 GUESS-ERROR: 2.309401 UPPER-LIMIT: 11 LOWER-LIMIT: 0)
(ACTUAL ANSWER IS 2)

26 – *(ANS '(AND (AND MALE LOCHOL) MEDTEMP) 'LEASTFREQ 'PREDNISONE)
= What was the frequency of the least common dosage?
(GUESS: 3.0 GUESS-ERROR: 1.328403 UPPER-LIMIT: 11 LOWER-LIMIT: 1)
(ACTUAL ANSWER IS 1)

[Again, note that estimates on sets involving three intersections are not as good as the corresponding estimates on sets involving two intersections.]

Chapter 3
Methods: the rules

And then there were the imaginary dragons, and the α-, anti-, and minus-dragons (colloquially termed nots, noughts, and oughtn'ts by the experts), the minuses being the most interesting on account of the well-known dracological paradox: when two minuses hypercontiguate (an operation in the algebra of dragons corresponding roughly to simple multiplication), the product is 0.6 dragon, a real nonplusser. Bitter controversy raged among the experts on the question of whether, as half of them claimed, this fractional beast began from the head down, or, as the other half maintained, from the tail up. Trurl and Klapaucius made a great contribution by showing the error of both positions. They were first to apply probability theory to this area and, in so doing, created the field of statistical draconics which says that dragons are thermodynamically impossible only in the probabilistic sense, as elves, fairies, gnomes, witches, pixies and the like.

Stanislaw Lem, Cyberiada, *1967, trans. M. Kandel (Avon, 1976)*

Our methods in this thesis fall into two categories, those for the rules and those for the database abstract. In this chapter we discuss the former, and in the next chapter the latter.

We shall present the rules according to the rule taxonomy sketched in section 1.4.3. As we said, there are four dimensions: (1) what statistic is being estimated; (2) what answer characteristic is being supplied by the rule (estimate, error, bound, or exact answer); (3) the situation in which the rule works (the "computational" dimension); and (4) what category of reasoning supports it (the "derivational" dimension). We discuss these dimensions in sections 3.2, 3.3, 3.5, and 3.6 respectively. Section 3.4 provides the necessary background on the algebra of quadruples for understanding the rule computations in section 3.5, and section 3.6 covers a few general issues concerning the rules.

As we said in section 1.4.1, queries consist of a statistical operator applied to three arguments, a relation, a query set (tuple set) on that relation, and a field (or virtual field) representing attributes of that set. In formal specification of rules in this chapter, where we omit the relation name or the field name they should be taken to be the same in all query expressions within a rule.

Appendix C provides concrete examples of the major rule categories to be discussed.

3.1. The statistic dimension of rules

3.1.1. The statistics examined

In this work we have explored "knowledge-based" reasoning from several rather different sources of knowledge. Thus in contradistinction to most work in statistics, we have deliberately addressed a variety of quite different statistical aggregates within the same research framework. See Figure 3-1. Formal definitions of the statistics are contained in Appendix A.

The basic such statistical aggregate operations, alias "statistics", with which we are dealing are number of items in a set (SIZE); the three classical measures of central tendency: MEAN, MEDIAN, and MODE; and the three classical measures of dispersion: MAX, MIN, and standard deviation[5], symbolized as SIGMA. These have had central attention in our work.

The obvious application of the above statistics is on the distribution of values of a numeric attribute of items in a database. But there are several derived distributions for which it is useful to tabulate additional statistics, giving them separate names. Three important such derived distributions are the frequency distribution of values by decreasing frequency, the distribution of gaps (spacings) between successive ordered values, and the fit of the original values to some ideal distribution. All three are useful in characterizing essential characteristics of attribute values not captured by the basic measures of central tendency and dispersion, but still very important tc statistical analysis, to varying degrees for different situations. Even when they are not directly of use, they are often helpful for estimating other more interesting statistics. Frequency distributions in particular are quite important because they can also characterize nonnumeric attributes.

For the first derived distribution, we have used statistics on the number of distinct values in a set (SIZEUNIQUE), the frequency of the mode (MODEFREQ), the frequency of the second most common item (MODEFREQ2), and the frequency of the least-commonly-occurring item that still has nonzero frequency (LEASTFREQ). For the second, we have used the maximum gap between distinct values (MAXGAP) and the minimum gap (MINGAP). For the third, we have used only an

[5]In this work we compute the standard deviation as $[\Sigma(x_i-\mu_x)^2/n]^{1/2}$ instead of the more usual $[\Sigma(x_i-\mu_x)^2/(n-1)]^{1/2}$, as it simplifies formulas. Because we only consider as meaningful those statistics on sets ten items or more, the discrepancy between the formulas is no more than about 5%.

Basic stats	Frequency stats	Gap stats	Fit stats
SIZE	SIZEUNIQUE		
MEAN			
MEDIAN			
MODE			
MAX	MODEFREQ	MAXGAP	MAXEVENDEV
	MODEFREQ2		
MIN	LEASTFREQ	MINGAP	MINEVENDEV
SIGMA			
KEYS			

Figure 3-1: The statistical aggregates studied in this work

even distribution to fit to, and the maximum (MAXEVENDEV) and minimum (MINEVENDEV) residuals from this fit. In addition to these things, we also tabulate for sets one more morsel of information, the fields and groups of fields which are "extensional[6] keys" for that set, i.e. for which every item is distinct in some particular database state.

3.1.2. Why histogram-statistic rules were not included

Clearly there are many other statistics we could study, as for instance other order statistics, proportions, other fits, and so on. We are only providing a framework for many possible future research efforts. In particular, we have chosen not to include the obvious possibility of histograms in characterizing distributions. There are several reasons for this:

- histograms must be in regard to a particular distribution, and there are four interrelated but distinct distributions of interest we have explored, plus a number of others. Good histograms for each would take up considerable space. We are particularly interested in supporting abstracts in which only, say, three words of storage are allocated to each set of interest, but where there are many such sets.

- histograms represent a systematic but "nonsemantic" representation of data, which ignores the fact that certain bins may be much more interesting than others. Carefully selected non-histogram statistics like those we are using can better reflect the particularly striking characteristics of data.

- much of the power of our reasoning, as will be seen, resides in the exploitation of multiple data dimensions. Histograms characterize only a single dimension well.

- we are likely to implicitly include histograms anyway in the database abstract, in the form of statistics on "first-order sets" representing partitions on the values of attributes, as we shall discuss in detail in section 4.2.2

[6] We shall use the terminology "extensional" to describe things like dependencies and key relationships between database entities that happen to be true only for a particular database state, instead of for any database state, the usual meanings of the terms. Our work is concerned with particular database states (extensions), not abstractions on all states (intensions) -- see section 3.4.5.2. For instance, if the ship-nationality attribute of a set of ships uniquely identifies ships in that set, it is considered an "extensional key" of the subrelation corresponding to that set, even though it is not a key in general (there are many ships of the same nationality in the database). On the other hand, ship-ID-number is an extensional key for any database set, and hence an unqualified key.

3.1.3. Why correlations rules were not included

We have chosen not to include in this analysis another obvious possibility, correlations between attributes (e.g., the linear correlation coefficient ρ), for several reasons:

- there are $O(N^2)$ binary correlations for N fields, as opposed to $O(N)$ of all the statistics mentioned above, hence they require a lot more space
- there are many different kinds of correlations: linear and nonlinear regression equations for numeric attributes, "generalized" tuples using higher-level symbols for all attributes, phenomena akin to database dependencies (functional dependencies, multivalued dependencies, join dependencies, etc.) but true only for a particular database state, and so on
- correlations are implicitly represented anyway by set-size statistics on the intersection of two simple sets (e.g. if the intersection is larger than expected by the product of the selectivities, the two intersected sets are positively correlated); an explicit correlation just is a kind of meta-statement about the statistics of a family of set intersections
- linear correlation between numeric fields can be derived from the MEAN of a TIMES of two fields, as explained below in 3.5.5.1
- correlations can be used in many different ways in rules, and using them effectively is a large undertaking

3.2. The answer dimension of rules

3.2.1. The form of answers

We give two kinds of answers to statistical queries, exact answers and bounded approximations. The former arise only occasionally, most notably for unions of disjoint sets. For instance, the size of a disjoint union is exactly the sum of the sizes of the component sets. Two more examples of exact rules: the maximum of a union of any two sets in the larger of the maxima of the two sets, and the mean of the sum of corresponding numeric attributes for the items of some set is the sum of the means of the set for each attribute.

If we cannot give an exact answer, we always give an estimate in the form of a quadruple (*always* a quadruple, never more nor less), which for statistics which are numbers (the predominant case) contains:

- a "best estimate" of the answer (alias EST)
- a standard error associated with that answer (alias ERR), i.e., the standard deviation associated with the EST estimate

- an absolute upper bound on the answer (alias SUP), a "less than or equal" bound[7]
- an absolute lower bound on the answer (alias INF), a "greater than or equal" bound[8]

For the few statistic values which are nonnumeric (e.g. the mode of a nonnumeric attribute, or the keys of a database relation) we have instead:

- a "best guess" as to the answer
- a value superset guaranteed to contain the answer

though we have now removed this from the implementation, after some experiments, as it requires a good deal of effort to define interactions with other uncertainty, while having low payoff.

3.2.2. Justification of quadruples

We arrived at the quadruple representation of query answers after a good deal of thought. Our basic objective was to find a robust way of characterizing uncertainty about a statistical value that did not require probabilities, certainty factors, or their ilk. These often require reasoning about subjective phenomena such as behavioral probabilities and possible worlds that are hard to pin down and inherently subject to expert biases and errors. On the other hand, absolute bounds on answers derived by logical deduction (our SUP and INF) cannot be argued. And expected values of the answer distribution and the deviations about its mean (the EST and ERR, respectively) substantially "integrate out" (in the calculus sense) subjectivity, reducing "noise" analogous to the way that capacitors (as integrators) are used to reduce electrical noise.

We believe we need all four of these entities to characterize uncertainty. An estimate isn't much help unless one knows how good it is, and the standard way to quantify this is a standard error; without this information the combining of estimates from different sources is virtually impossible. Absolute bounds on an estimate can be quite helpful to analysis, though statisticians have not been much concerned with them (though there are exceptions, e.g. results related to Chebyshev's inequality like [Fahmy and Proschan 81, Arnold 74]). Bounds (also under the name "constraints") are a central issue in much of computer science, and their manipulation the source of

[7]Do not confuse the SUP upper bound with the MAX (maximum) statistic, which is quite different. MAX applies to a finite, knowable-in-principle set of values contained in a database. SUP applies to a distribution of a subjective random variable, the possible values that a particular *statistic* on data values can take on.

[8]Similarly, don't confuse INF with the MIN (minimum) statistic.

many major accomplishments. Bounds on computation time and space are important for analysis of algorithms (e.g. [Graham, Yao, and Yao 80]). Artificial intelligence in particular has exploited novel methods of problem-solving based on nonnumeric-constraint propagation. Constraints can be used to simplify search problems: if you absolutely know that an answer can't be of a certain kind, you can prune the branch corresponding to it. Even when constraints are weak, the cumulative effect of many can be important, as in many linear programming problems near the optimum point.

3.3. Algebra of quadruples

Before going on to discuss the computational dimension of rules, we must articulate what the computations *are*. Most of our rules involve mathematical operations on the values of certain statistics to get certain other desired statistics. When the former are expressed as quadruples we need a special algebra to define the results for this special data type.

3.3.1. Bounds: interval mathematics

Interval mathematics [Rall 81, Cole and Morrison 82] is an offshoot of numerical analysis concerned with analyzing propagation of absolute errors in numerical calculations. We can use it to determine the effect of numerical operations on bounds. Denoting the closed interval from the number a to the number b as [a,b] (that is, INF is a and SUP is b) we can give the laws of interval mathematics for the arithmetic operations important in our system as follows:

$$[a,b] + [c,d] = [a+c,b+d]$$

$$[a,b] - [c,d] = [a-c,b-d]$$

$$\begin{aligned}[a,b] * [c,d] &= [a*c,b*d] \text{ if } a \geq 0 \text{ or } c \geq 0\\ &= [b*d,a*c] \text{ if } b < 0 \text{ and } d < 0\\ &= [\min(a*c,b*c,a*d,b*d),\max(a*c,b*c,a*d,b*d)] \text{ otherwise}\end{aligned}$$

$$\begin{aligned}[a,b] / [c,d] &= [a/d,b/c] \text{ if } c > 0\\ &= [b/c,a/d] \text{ if } d < 0\\ &= [-\infty,a/c] \cup [b/d,+\infty] \text{ if } a \geq 0 \text{ and } c \leq 0 \leq d\\ &= [-\infty,a/d] \cup [b/c,+\infty] \text{ if } b < 0 \text{ and } c \leq 0 \leq d\end{aligned}$$

$$\max([a,b],[c,d]) = [\max(a,c),\max(b,d)]$$

$$\min([a,b],[c,d]) = [\min(a,c),\min(b,d)]$$

$f([a,b]) = [f(a),f(b)]$ where f is a monotonically increasing function on [a,b]

(important examples of f being log, antilog, square, square root)

$$\text{abs}([a,b]) = [a,b] \text{ if } a \geq 0$$
$$= [-b,-a] \text{ if } b < 0$$
$$= [0,\max(-a,b)] \text{ otherwise}$$

Division by an interval containing zero is the most serious problem in the above, since the result is two disjoint and unbounded intervals, and cannot be expressed as in our quadruples. (We ignore this case in our implementation.) Sometimes calculations can be rearranged to eliminate division by such an interval, a trick used by numerical analysts.

Comparisons (< and >) between intervals are possible, but must set to "undefined" for cases where the ranges of the intervals overlap. Formally:

$[a,b] < [c,d]$ iff $b<c$;
$[a,b] > [c,d]$ iff $a>d$;
undefined otherwise

All these rules assume independence of the random variables representing distributions, and if this is not the case better (tighter) ranges can be found, essentially by defining a new function and performing an analysis of it. For instance, to use the example of [Cole and Morrison 82], $f(x) = x+(K/x)$ is monotonically increasing for $x>\sqrt{K}$ because $df(x)/dx = 1 - (K/x^2)$ is greater than 0 for $x>\sqrt{K}$. Hence f on the interval [a,b] has a minimum at f(a) and a maximum at f(b), provided $\sqrt{K}<a$. Thus

$$SUP(x+(K/x)) = SUP(x) + K/SUP(x)$$
$$INF(x+(K/x)) = INF(x) + K/INF(x)$$

as long as $INF(x) > \sqrt{K}$, which would not be true if the occurrences of x were not linked, since by our algebra

$$[a,b] + K/[a,b]) = [a,b] + [K/b,K/a] = [a+K/b,b+K/a]$$

or in other words:

$$SUP(x+(K/y)) = SUP(x) + K/INF(y)$$
$$INF(x+(K/y)) = INF(x) + K/SUP(y)$$

which will always be worse (broader) bounds than those obtained from the derivative reasoning.

Note that exact numbers (as opposed to ranges) are perfectly compatible with the above algebra; just treat them as pairs where the upper bound equals the lower.

3.3.2. Estimates and errors: normal curve algebra

For the estimates (ESTs) and standard errors (ERRs) of quadruples we need a quite different kind of algebra, modelling the quadruples as denoting truncated normal distributions and combining them according to standard probability theory. It can be proved from information theory that the least-information-content probability distribution one can construct given a maximum, minimum, mean, and standard deviation is a truncated normal one, and that is why we make this assumption[9]. It can be shown in turn, on the basis of a number of sophisticated criteria, that the least-information-content distribution is the most "reasonable" one to make when one wishes to best summarize a partial degree of knowledge; any other will get you into certain contradictions [Shore and Johnson 81].

There is in addition both experimental and other theoretical justification for such a distribution from sampling methods. Even if a distribution is highly skewed or otherwise far from a normal distribution, sampling distributions of its statistics (e.g. mean, median, standard deviation, maximum) tend to be normal. Sampling is an important concept since set intersections are a kind of mutual nonrandom sample.

Our algebra for ESTs and ERRs is that for expected values of normal random variables selected without replacement from their distributions used in [Haugen 68]. Unfortunately however, the rules are only approximations, not absolutes as with bounds. Denote an EST-ERR pair as <a,b>. Then:

$$\langle a,b\rangle + \langle c,d\rangle \approx \langle a+c, \sqrt{(b^2 + d^2)}\rangle$$

$$\langle a,b\rangle - \langle c,d\rangle \approx \langle a-c, \sqrt{(b^2 + d^2)}\rangle$$

$$\langle a,b\rangle * \langle c,d\rangle \approx \langle a*c, \sqrt{(b^2c^2 + a^2d^2)}\rangle$$

$$\langle a,b\rangle / \langle c,d\rangle \approx \langle a/c, b/\mathrm{abs}(c) + \mathrm{abs}(c)/d\rangle$$
for abs(a) significantly > b, abs(c) significantly > d

$$f(\langle a,b\rangle) \approx \langle f(a),(f(a+b)-f(a-b))/2\rangle$$
for f continuous and abs(a) significantly > b (see section 3 5.1.4)

The first three results can be derived from computing the cumulative distribution of the result (the

[9]Strictly speaking, the distribution is a truncation of $c^{a+bx+cx^2} = Ac+[(x-C)^2]$, or a displaced normal curve. In some cases (to be discussed in section 3.4.1) the standard deviation may not be known, and the maximum-entropy distribution is a truncated exponential; and in others, only the bounds are known, and the maximum-entropy distribution is uniform. These can correspond to c=0 in the first case, b=0 and c=0 in the second.

probability that the result is less than some fixed number x_0) and taking the derivative of this with respect to x_0. Unfortunately, there is no nice formula for the distribution of the maximum or minimum of two random variables[10]; we use a crude fitting of two appended uniform distribution to get an answer in our implementation.

Again, note that exact numbers (instead of <EST,ERR> pairs) are perfectly compatible with the preceding algebra; they're just things with zero ERR.

3.3.3. Combining evidence

There is one other operation on quadruples of great importance to our system, namely the combining of evidence from the results of multiple rules applicable to the same situation. We gave the formulas for this in section 1.6.1. As with the rest of quadruple algebra, the combined bounds are exact but the estimate and error only approximate. It is difficult to know how much weight to give to different EST and ERR rules applying to the same situation, an important issue since some ways of obtaining ESTs and ERRs are much better than others in a particular situation. For this reason we try to ensure that only one EST rule and one ERR rule apply to a query, while making no attempt to control the number of the bounds rules that apply. We do this by enforcing a priority on EST and ERR rules where a more specific rule always preempts a more general one. In the two-level case, this becomes a scheme for rule defaults. We do not consider more sophisticated rule selection architectures, though there is much room for further research in this direction.

3.3.4. Embedded quadruple algebra

Early in this work we felt it necessary to allow for indefinite embedding within quadruples. For instance, we could characterize a lower bound by its own estimate, standard error, and bounds. The idea was that absolute bounds can get to be quite pessimistic in many calculations, and perhaps a lower bound on an upper bound, or an upper bound on a lower bound, is needed to illustrate just

[10]Denote two normally distributed random variables as $N_1(x)$ and $N_2(x)$, and their integrals (erf(x)'s) as $\Phi_1(x)$ and $\Phi_2(x)$. Then the distribution of the maximum is the derivative of $\Phi_1(x)\Phi_2(x)$, and the mean is

$$\begin{aligned}&\int x \, d[\Phi_1(x)\Phi_2(x)]/dx \\ &= x\Phi_1(x)\Phi_2(x) - \int \Phi_1(x)\Phi_2(x)\, dx\end{aligned}$$

from integration by parts, and the latter integral has no closed-form solution. But we can get an approximation by expressing this function as

$$N_1(x)\Phi_2(x) + \Phi_1(x)N_2(x)$$

and using numerical methods to approximate its mean.

how broad the bounds were. All the abovementioned algebra can be extended to embedded quadruples by introducing recursion appropriately. For instance:

```
[[a,b],[c,d]] + [[e,f],[g,h]] = [[a,b]+[e,f] , [c,d]+[g,h]]
                              = [[a+e,b+f],[[c+g,d+h]]
```

When embedded quadruples become too far embedded to make computation efficient, it is helpful to apply a simplification operation to restore them to a single-level form, defined as follows:

```
SIMPLESTAT(QUAD) =
  LIST(SIMPLESUP(QUAD),SIMPLEINF(QUAD),SIMPLEEST(QUAD),
    SIMPLEERR(QUAD))
SIMPLESUP(QUAD) = QUAD if QUAD is an atom,
    SIMPLESUP(CAR(QUAD)) otherwise
SIMPLEINF(QUAD) = QUAD if QUAD is an atom,
    SIMPLESUP(CADR(QUAD)) otherwise
SIMPLEEST(QUAD) = QUAD if QUAD is an atom,
    SIMPLESUP(CADDR(QUAD)) otherwise
SIMPLEERR(QUAD) = QUAD if QUAD is an atom,
    SIMPLEEST(CADDDR(QUAD)) otherwise
```

(Note that last SIMPLEEST is not a misprint.)

However, we now question the value of embedded quadruples. They make the algebra much slower, are difficult to explain and debug, and rarely provide much advantage.

3.3.5. Integer quadruples

If we know a quadruple represents something that is an integer (for instance, if it is a count of a set), we "integerize" it by mapping the SUP to its floor and the INF to its ceiling. Our considered judgment was that nothing special should be done to the EST and ERR in this circumstance because useful information might be lost. While it sounds funny to say that the average American family has 2.3 children, it does convey a sense of how much more frequent a 2-child family is compared to a 3-child family.

3.4. The computational dimension of rules

The query argument dimension concerns the nature of the query submitted to the system. There are three basic kinds, or "levels" of such rules: (1) intraquadruple rules, (2) rules depending only the statistic queried, and (3) rules that examine the query set and query attribute as well as the statistic. There are just a few of the first, a number of the second, and quite a lot of the latter.

Section 3.4.1 covers the intraquadruple rules, and 3.4.2 the statistic-specific. The next two sections cover two very important classes of rules based on query expression decomposition: set decomposition in 3.4.3, and attribute decomposition in 3.4.4. 3.4.5 covers some rather different rules suggested by artificial-intelligence concept of inheritance; 3.4.6 rules for rearranging queries to logically equivalent forms; and 3.4.7 some miscellaneous (but important) rules.

3.4.1. Intraquadruple rules

With the algebraic manipulations of the last section, there is no guarantee that the EST and ERR will remain consistent with the SUP and INF bounds in a quadruple. For instance, the mean American ship may be 300 feet long, the mean tanker 600 feet, and a reasonable EST of the mean American tanker length can be the average of the two, or 450 feet. But it may possibly be the case that American ships only run 50 to 400 feet, hence the longest one is no more than 400, and hence the mean for any subset cannot be larger than 400.

There are also many situations in which good bounds are available, but no good ways of getting EST and ERR values (as in reasoning about the sizes of sets based on mode frequencies and least-common-item frequencies), and many other situations where EST is known but it is hard to get the associated ERR (like the rule that a good estimate for the mode of a subset is the mode of the set). Thus we may need to compute a "reasonable" EST and ERR from a SUP and INF to supply to the user.

Our policy is to assume primacy of the SUP and INF bounds to the estimate (EST), and this in turn to the error (ERR). SUP and INF are always true, assuming no bugs in an implemented system that computes them, and hence form an "anchor" for obtaining the unguaranteed EST. ERR only makes sense associated with an EST, as a kind of qualifier, and hence should be estimated from it.

As for specific details: if the EST is not in the range SUP to INF, or there is no estimate whatsoever, the EST is set to the midpoint of the range, the average of SUP and INF. (This is the value corresponding to the maximum entropy distribution given knowledge only of maximum and minimum, the even distribution.) This done, the ERR is then examined. If it is greater than the geometric mean of the deviations of EST from SUP and INF, or if there is no ERR, it is set to that same geometric mean divided by the square root of 2. (We justify this as the "midpoint variance" between the maximum variance and zero; unfortunately the maximum-entropy value for the standard deviation cannot be computed in closed form[11].) As for SUP and INF, if either is missing (though this happens very rarely), they are set to $+\infty$ and $-\infty$ respectively (approximations, actually). If SUP=INF, the quadruple is replaced by that single number, and EST and ERR ignored.

3.4.2. Rules relating statistics

The next kind of rules are those relating statistics on the same set and attribute to one another. Most of these are bounds rules. For example, an upper bound on the mean is the maximum; the mean is never more than one standard deviation away from the median; and mode frequency is never less than the number of items in the set divided by the number of distinct items.

These rules can generally be used in several directions. For instance, the latter example can be inverted in either of two ways, so we have three results:

$$\text{MODEFREQ} \geq \text{SIZE} / \text{SIZEUNIQUE}$$
$$\text{SIZEUNIQUE} \geq \text{SIZE} / \text{MODEFREQ}$$
$$\text{SIZE} \leq \text{SIZEUNIQUE} * \text{MODEFREQ}$$

and we can use whichever version we need at some point, depending on the degree of knowledge

---

[11]Note

$$\int x e^{\lambda x}\, dx = e^{\lambda x}(\lambda x-1)/\lambda^2$$

and the mean of the general exponential probability distribution $p(x) = \lambda e^{\lambda x}/[e^{\lambda b}-e^{\lambda a}]$ on the interval a to b is

$$\mu = [e^{\lambda b}(\lambda b-1) - e^{\lambda a}(\lambda a-1)] / \lambda^2[e^{\lambda b}-e^{\lambda a}]$$

which, given the constant mean μ it is equal to, cannot be solved in general for λ except numerically. (The scaling of $p(x)$ is the reciprocal of the integral of $e^{\lambda x}$ over the interval, or $[e^{\lambda b} - e^{\lambda a}] / \lambda$.) Then once λ is found the distribution in fully parameterized, and its standard deviation can be computed from the formula

$$\sigma^2 = -\mu^2 + \int x^2 \lambda e^{\lambda x}/[e^{\lambda b}-e^{\lambda a}]\, dx$$
$$= -\mu^2 + [e^{\lambda b}(b^2 - 2b/\lambda + 2/\lambda^2) - e^{\lambda a}(a^2 - 2a/\lambda + 2/\lambda^2)] / [e^{\lambda b}-e^{\lambda a}]$$

If however the mean is near the middle of the range between the maximum and the minimum, λ will be small, and we can get a closed-form approximation using $e^{\lambda x} \approx 1 + \lambda x$. This means then that

$$p(x) \approx [1 + \lambda x] / [(b-a) + .5\lambda(b^2-a^2)]$$

so using the formula

$$\int x\,[1 + \lambda x]\, dx = .5x^2 + .3333\lambda x^3$$

the mean is approximated by

$$\mu \approx [.5(b^2 - a^2) + .3333\lambda(b^3 - a^3)] / [(b-a) + .5\lambda(b^2-a^2)]$$
$$= [.5(b+a) + .3333\lambda(b^2 + ab + a^2)] / [1 + .5\lambda(b+a)]$$

$$\lambda \approx [\mu - .5(b+a)] / [.3333(b^2 + ab + a^2) - .5\mu(b+a)]$$

Again, this approximation requires small λ.

about these values. If we are not careful with this, however, we can get into infinite cycles during rule application. To avoid this we must designate some rules as "forwards" and other as "backwards" and treat them differently; see section 5.5 for further discussion. (Do not confuse the forwards-backwards distinction with the upwards-downwards-lateral-diagonal distinction forthcoming in section 3.4.5.)

Rules relating statistics are very important because there are many queries for which "standard" rules involving decompositions, to be discussed in the rest of this elaboration of the query argument dimension, do not exist. Just as with integration in symbolic algebra systems [Tobey 71], there turn out to be some situations which have closed form "solutions" (i.e., query rearrangements), and others without. For instance, there is a nice rule for the median of the union of two disjoint sets (it is bounded by the medians of the two sets), but no rule at all for the median of the intersection. But our domain has one advantage that symbolic algebra does not: we give estimates and bounds usually, not exact answers, and weak general-purpose statistic-statistic rules can always be used to get such information.

3.4.3. Set expression decomposition

There are four categories for decomposition of the set argument in a query: a simple parent set, set intersection, set union, and set complement.

3.4.3.1. Simple parent-child inheritance

In creating the database abstract we have room only for a certain number of partitions of the values of attributes. All other partitions we must relate to these. For instance, if ship statistics are tabulated by length ranges 100-to-200 feet, 200-to-300 feet, and so on, and if we wish to know something about ships that are 320 feet in length to 390 feet, we must relate it to the parent superset of ships 300-to-400. So an upper bound on the number of 320-to-390 ships is the number of 300-to-400, and an estimate is (390-320)/100 = 70% of that number; an upper bound on the maximum tonnage is the maximum tonnage of 300-to-400; and the mode nationality of 320-to-390 is probably the mode for 300-to-400. Artificial intelligence calls this "downwards inheritance"; see section 3.4.5.

A very important application of these rules is to the set of all items in a relation, what we call the "universe" set. Since it is the parent of every set on that relation, it can always be used with these superset-to-set rules. This is helpful for getting bounds, but not it is not usually as useful with estimate and error rules since the universe is usually a very much larger set than a random set on the relation, and can display quite different tendencies.

3.4.3.2. Set intersections

A set which is the intersection of two or more sets (which we denote AND(A,B) for two, AND(A,AND(B,C)) for three, etc.) is a special but very important case of the above. In this case there are two or more parents bestowing characteristics to the offspring. We must use the formulas for combining evidence (see 1.6.1). We believe that set intersections will be the most common type of construct in queries. It is very natural to specify a set by a series of restrictions that must be taken together, much more so than to specify by a disjunction of restrictions, or the opposite of a restriction. We have thus paid special attention to handling intersections well in our system, and have developed a number of apparently new methods for estimating statistics on them, as for instance items 5 through 9 of the set-size query processing example in section 1.5.1.

Note that set intersections can occur simultaneously with parent-sibling inheritance, and one must be careful to identify the immediate parents. For instance, suppose we wish to know something about the set of tankers in the Adriatic, and the database abstract only includes information about the set of tankers and the set of ships in the entire Mediterranean. We can consider this set as having only two immediate parents, the set of ships in the Adriatic, and the set of *tankers* in the Mediterranean. All other potential parents can be considered ancestors of these; for instance, the ships in the Mediterranean are an ancestor of ships in the Adriatic, and the tankers are an ancestor of the tankers in the Mediterranean.

3.4.3.3. Set unions and complements

There are two other simple set operations, union (OR) and complement (NOT). We need rules for each. Generally, unions tend to support good approximations, better than intersections, and complements tend to support approximations worse than either. Note also that any set complement can be written as a union of other sets, provided (as we shall always assume in this thesis) that the first-order sets representing partitions of a single attribute's values are disjoint; for instance, the set of patients without high temperatures is the union of the sets of patients with normal temperatures and the patients with low temperatures. Note also that:

OR(A,B) = NOT(AND(NOT(A),NOT(B)))

so unions may be substituted for by intersections and complements, providing another perspective.

By the way, set difference (i.e., AND(A,NOT(B))) is easy to define as a complement in which the universe is some set A and not the whole database as in a complement. The same rules for complement apply.

3.4.3.4. Special cases

There are three important special cases for set-expression decomposition that must be recognized when they occur: (1) disjoint sets, (2) sets that are subsets of others, and (3) "covering" sets, sets that include every item on a particular range. We can write the general rules as follows:

AND(A,B) = the null set if A and B are disjoint
AND(A,B) = A if A is a subset of B
= B if B is a subset of A
OR(A,B) = B if A is a subset of B
= A if B is a subset of A

There are also rules for specific statistics, for instance:

SIZE(OR(A,B)) = A + B if A and B are disjoint
SUP(MEDIAN(OR(A,B))) = LARGEROF(MEDIAN(A),MEDIAN(B))
A and B are disjoint
INF(MIN(NOT(A))) = MAX(A) if A is a cover on the attribute referred to,
and MIN(A) = MIN(UNIVERSE)

These conditions are checked for first in processing a query, before any other rules are tried. If they can be found to apply, a query can be simplified, and the accuracy of its answer can be greatly improved or even an exact answer supplied. Here is the formal definition of what the rules do. Let DISJOINTP(A,B) be the predicate that A and B are disjoint sets, let SUBSETP(A,B) mean whether A is a subset of B, and let COVERP(A,F) mean that A includes every item in the universe set that has a value in that range on attribute F. The three conditions can be defined as follows:

DISJOINTP(A,B) = T if A and B are simple partitions on the values of some same field
DISJOINTP(A,B) = DISJOINTP(B,A)
DISJOINTP(A,B) = T if MAX(A) < MIN(B) or MAX(B) < MIN(A)
for some numeric attribute
DISJOINTP(AND(A,B),C) = T if OR(DISJOINTP(A,C),DISJOINTP(B,C))
DISJOINTP(OR(A,B),C) = T if AND(DISJOINTP(A,C),DISJOINTP(B,C))

SUBSETP(A,B) = NIL if A and B are simple partitions on the values of the same field
SUBSETP(A,B) = T if A = B
SUBSETP(AND(A,B),C) = T if AND(SUBSETP(A,C),SUBSETP(B,C))
SUBSETP(OR(A,B),C) = T if OR(SUBSETP(A,C),SUBSETP(B,C))
SUBSETP(A,AND(B,C)) = T if OR(SUBSETP(A,C),SUBSETP(B,C))
SUBSETP(A,OR(B,C)) = T if AND(SUBSETP(A,C),SUBSETP(B,C))
SUBSETP(NOT(A),B) = DISJOINTP(NOT(A),NOT(B))
SUBSETP(A,NOT(B)) = DISJOINTP(A,B)

COVERP(A,F) = T if A is defined as a partition on the values of attribute F
COVERP(AND(A,B),F) = T if AND(COVERP(A,F),COVERP(B,F))
COVERP(OR(A,B),F) = T if AND(COVERP(A,F),COVERP(B,F))
COVERP(NOT(A),F) = T if COVERP(A,F)

Note that while these rules may not apply often in simple queries, they will occur in more complicated queries, especially when the same set is referred to more than once, and they can provide a much better answer when they can be applied.

3.4.4. Attribute (field) expression decomposition

Just as sets may be operated on by intersections, unions, and complements, the attributes may be combined in various ways to get virtual attributes. Rules can decompose these constructs analogously.

3.4.4.1. Unary operations on fields

Unary operations applied to numeric attributes generally support good inferences, and particularly so when the functional operations are continuous and monotonic in the interval of interest. For instance, the maximum of the logarithms of an attribute is the logarithm of the maximum; the median of the logarithms is the logarithm of the median; and the number of distinct values of the logarithms of an attribute is the number of distinct values of the attribute, provided no two numbers are closer than the rounding error in the calculation. But the logarithm of the mean is only an upper bound on the mean of the logarithms (its true value is the logarithm of the geometric mean of the original attributes), and its standard deviation is similarly hard to analyze. The rules for logarithm are similar to those for antilog, square root, square, and reciprocal, all commonly used functions in statistical analysis. We have also looked at a few nonmonotonic functions, e.g. absolute value, which can be seen as an upwards shift of some of the values of attributes of a range.

Unary functions, of course, have been subject of a good deal of attention throughout the history of mathematics, much of it quite simple and exploitable for our needs. For instance, for any function whose second derivative is constant in sign (as log, antilog, square root, square, and reciprocal are for positive numbers), tangent lines form bounds on the curve (upper bounds if the second derivative is negative, lower bounds if it is positive). Secants of the curve across some range may be used to bound in the other direction. These linear bounding functions (for which calculation of statistics is much easier) can give quite tight bounds, as described further in section 3.5.1.4.

Nonnumeric attributes are also subject to functional transformations. For instance, the nationalities of ships may be generalized to regions of the world, and statistics estimated on, say, the European ships. We specify all such mappings in a common form, as an association list of mapped-to symbols and the lower-level symbols which map to them. We then use the association list in making estimates of statistics on the generalized values. For instance, an upper bound on the mode frequency of the generalized values is the mode frequency of the original values times the size of the largest set in the association list. Note that these operations go in one direction only: given data of ship regions-of-nationality, one cannot reconstruct the nationalities. Thus it is important in setting up the database abstract to have statistics on a relatively "low level" of attributes.

3.4.4.2. Binary operations

One can also have derived attributes from more than one attribute with respect to some set. For instance, one may be interested in mean of the values created by adding the corresponding values for loading time of a ship in port and its voyage duration. We only consider binary operations here because other operations can almost always be decomposed into binary ones; see section 5.6.4.

The algebra of quadruples discussed in 3.3 can substantially be used for getting such statistics. But there is one subtle difference for estimate and error rules: *the random variables represent drawings without replacement from a population.* So for instance, the mean of the sums of the corresponding elements of two fields is exactly the sum of their means, instead of just being approximated by it, because:

$$\Sigma(x+y)/n = \Sigma x/n + \Sigma y/n$$

Most of the other rules look the same, however.

These arithmetic operations have surprisingly poor absolute bounds for frequency distribution statistics (number of distinct items, mode frequency, etc.) This is because it is possible, for instance, for many different number pairs to add up to the same total, and same-total coincidence pairs can always arise unexpectedly, if not by some underlying phenomenon in the data. But this happens infrequently, and we are usually safe in ignoring them for computing the estimate (EST).

3.4.4.3. The vectorization operation

There's one very important nonnumeric binary operation we haven't mentioned, the creating of vectors from corresponding values of two attributes. For instance, we can can vectorize the ship attributes of current latitude and current longitude into a ship-position virtual attribute, and talk about about statistics on ship positions. Note that this operation creates a nonnumeric attribute, whether or not the component attributes are nonnumeric. Note also this operation provides a definition of an "extensional key" of a relation at a given time: any set of attributes which when vectorized, have a mode frequency statistic equal to 1. Vectorization is what [Smith and Smith 77] and other database work has called "aggregation", but that word is usually used in statistical computing to mean calculation of statistics like count, mean, median, etc. on sets, and we shall avoid it.

3.4.4.4. Operations with constants

Operations can also take place with respect to constants (or, strictly speaking, attributes whose values are all some constant). Rules are particularly simple for these cases, and almost all exact. For instance, for K some real number:

MEDIAN(Q,TIMES(K,A)) = K*MEDIAN(Q,A)
SIGMA(Q,TIMES(K,A)) = K*SIGMA(Q,A)

3.4.5. Artificial-intelligence inheritances

Another very different class of rules along the query argument dimension originate from the artificial-intelligence concept of inheritance. Statistical aggregate properties such as mean, maximum, and mode have not previously been thought to "inherit" between sets (cf. [Fahlman80], which uses the claim that inheritance of normative and prototype properties can't be done between two types or sets to justify the distinction between individuals and types). But they do in a weak sense, and a collection of such "weak" information can often be combined to get stronger information. We have explored these sorts of rules only partially in our implementation.

3.4.5.1. Some motivation for inheritances

Inheritance provides a different metaphor for rules than the algebraic style we have thus far considered. Consider the following example. Suppose we have conducted a census of all elephants in the world and we can definitely say that all elephants are gray. Then by set-to-subset inheritance of the "color" property, the set of elephants in Clyde's herd must be gray, Clyde's herd being some particular herd of elephants.

This will not work for statistical aggregate properties such as maximum and mean. Suppose our census found that the longest elephant in the world is 27 feet long, and the average elephant 15 feet. This does *not* mean the longest elephant in Clyde's herd is 27 feet, nor the average in the herd 15 feet. But a weak form of inheritance is present, for we can assign different degrees of likelihood to the following:

1. "The longest elephant in Clyde's herd is 30 feet long."
2. "The average elephant in Clyde's herd is 30 feet long."
3. "The longest elephant in Clyde's herd is 27 feet long."
4. "The average elephant in Clyde's herd is 27 feet long."
5. "The longest elephant in Clyde's herd is 16 feet long."
6. "The average elephant in Clyde's herd is 16 feet long."

Statements 1 and 2 are impossible. Statement 3 is possible but a bit unlikely, whereas statement 4 is almost certainly impossible. Statement 5 is surprising and hence apparently unlikely, whereas 6 is quite reasonable. Since we don't know anything of Clyde's herd other than that they are elephants, a kind of inheritance from the properties of elephants in general must be happening.

3.4.5.2. Our four-characteristic approach

Inheritance phenomena can take place with any statistic, and form the basis of a complete theory of knowledge representation. To site our approach relative to other knowledge representation work:

- our theory concerns set representation only (but sets of SIZE=1 can represent individuals)
- it concerns "definitional" sets primarily (those with absolute criteria for membership) as opposed to "natural kind" sets [Brachman and Israel 81] (though degrees of set membership as in fuzzy set theory could be introduced)
- it mainly deals with extensions (exemplars), not intensions (meanings); or in other words, it deals with a particular database state and not things with are true for any valid database state, like the "extensional keys" mentioned in the footnote in section 3.1
- it only addresses the set-subset semantic relationship; however, often other relationships can be viewed this way by "atomization" of the included concepts, e.g. geographical containment may be seen as a set-subset relationship between sets of points

The key idea is to note that while in a few cases statistical properties inherit values exactly from set to set, in most cases they do not; but that there are characterizations of a numeric statistic that will inherit much more often than nonstatistical properties, namely bounds, estimates, and standard errors. Some examples:

- An upper bound on the mean of a subset is the maximum of the set.
- A lower bound on the maximum of a subset is the minimum of the set.
- A best estimate of the mean of a subset, in the absence of further information, is the mean of the set.
- The "sampling theorem of the mean": a standard deviation of the mean of a subset is approximately the standard deviation of the set times the square root of the difference of the reciprocals of the subset size and set size (i.e. ERR(MEAN(Q,F) is $\sigma(1/N - 1/n)$, where σ is the standard deviation of the set, n its size, and N the size of the subset)

The last also illustrates an important feature of statistical property inheritance, namely that functions (in the mathematical sense) of values may be inherited rather than the values themselves. But since the different values are so strongly coupled it seems fair to still call it "inheritance".

3.4.5.3. Inheritance types

The main categories of statistical inheritance are:

- Downwards inheritance. That is, from set to subset, as most of the previously mentioned rules. This is the usual direction for statistical inheritance since it is usually the direction of greatest fanout: people tend to store information more for general concepts than specific concepts, for broadest utility. In particular, downwards inheritance from sets to their intersection is very common in human reasoning, much more so than reasoning with unions and complements of sets. Downwards inheritance works best for statistics that represent a kind of universal quantification over all members of a set, e.g. set maximum.
- Upwards inheritance. Inheritance from subset to set occurs with set unions, in particular unions of disjoint sets which (a) seem easier for humans to grasp, and (b) have many nice inheritance properties (e.g. the largest elephant is the larger of the largest male and largest female elephant). Sampling, random or otherwise, to estimate characteristics of a population is another form of upwards inheritance, though with the special disadvantage of involving a non-definitional set. Upwards inheritance also arises with caching [Lenat et al 79] (see 5.4); people may cache data on some small subsets important to them (like Clyde's herd) in addition to general-purpose data. Upwards (as well as downwards) inheritance is helpful for dealing with "intermediate" concepts above the cache but below general-purpose knowledge (e.g. the set of elephants on Clyde's rangelands). Upwards inheritance works best of statistics that represent a kind of existential quantification over all members of a set, e.g. the possible values that items can have. We have not implemented arbitrary cache-based upwards inheritance for our

system, however, because it is a relatively weak inference and does not seem cost-effective unless a good deal of information is cached.

- Lateral inheritance. A set can suggest characteristics of sibling sets of the same parent superset [Carbonell 80]. Two examples are set complements (i.e. the set of all items not in a set, with respect to some universe), and when sibling sets differ only by an independent variable such as time or space, and there are constraints on the rate of change (i.e. derivatives) of numeric attributes between siblings (e.g. the stock market average on successive days). Lateral inheritance is much more domain-dependent than the other kinds of inheritance, because a set can have many siblings, and some kind of domain knowledge is needed to choose the single "best" one to inherit from. Unfortunately, except for some unusual situations, no bounds are possible with lateral inheritance, only estimate and error information. For these reasons we have not implemented general lateral inheritance in our system, only the complement usage.

- Diagonal inheritance. An interesting hybrid of downwards and lateral inheritance is possible with statistical properties. Given statistics on the parent and some set of siblings, we can often "subtract" out the effect of the known siblings from the parent to get better estimates on the unknown siblings. For instance, the number of female elephants is the total number of elephants minus the number of male elephants; and the mean of the length of female elephants is

  [MEAN(ALL)*SIZE(ALL) - MEAN(MALES)*SIZE(MALES)]
  / SIZE(FEMALES)

  This idea also works for moment and extrema statistics. We have not implemented this inheritance in our system because our abstracts had too many first-order set partitions per attribute to make these inferences very strong.

- Attribute-hierarchy inheritance. A different kind of inheritance hierarchy arises with attribute vectorizations (see 3.4.4.3) on the same set, and there can be downwards, upwards, and lateral inheritances of this kind too. For instance, the number of distinct latitude-longitude pairs for a set of ships is lower-bounded by the number of distinct latitudes, a different form of downwards inheritance; and is upper-bounded by the number of distinct latitude-longitude-status triples. We have implemented the upwards version in our system, but neither of the others because we felt they would only arise for queries very awkward to phrase.

- Inheritance-rule inheritance. Some sets are sufficiently "special" to have additional inheritance rules for all subsets or supersets. An example is an all-integer set, where for any subset an upper bound on the number of distinct values for that property is the ceiling on the range. Since this is definitely a rule concept at work here, and one sufficiently different (and stronger) than other rules applying to these statistics, it make sense to designate it as a special rule rather than as the value of a special "integerness" statistic. We have not implemented this in our system as it requires theorem-proving reasoning.

3.4.6. Query rearrangement rules

Our three set operations of intersection, union, and complement are equivalent in power to the propositional calculus, and obey equivalents to all its theorems, as for instance associativity, absorption, and distributive laws. Virtual field expressions can also be rearranged.

3.4.6.1. Associativity

Commuting the order of a binary intersection or union should not affect the answer given by our system; we have tried to be careful about ensuring this in our implementation. But one of the disadvantages of requiring such operations to always be binary is that associativity does not hold, e.g.

SUP(SIZE(AND(A,(AND(B,C))))) is not necessarily SUP(SIZE(AND(AND(A,B),C))))

because analysis of the set B with respect to the attribute on which C is defined may give a tight bound which will not be noticed in the second case. (This rule is discussed in more detail in 3.4.7.2.) We try to get around this for intersections of three and four sets by defining all permutations, trying each separately, and combining the results. But things get rapidly out of hand as the number of sets increases, and we see no nice solution.

3.4.6.2. Absorption laws

One class of equivalent forms exists, however, where one form is definitely superior to the other, the so-called absorption-law forms, e.g.

OR(A,NOT(A)) = UNIVERSE
AND(A,NOT(A)) = the empty set
OR(A,AND(A,B)) = A
AND(A,OR(A,B)) = A
OR(A,AND(NOT(A),B)) = OR(A,B)
AND(A,OR(NOT(A),B)) = AND(A,B)

Here the second form is definitely preferred since it reduces the number of binary combinations and hence opportunities for inaccuracy in estimation. A general heuristic we suggest is that the fewer terms an equivalent expression has, the better it is. This is because our rules (for all of SUP, INF, EST, and ERR) implicitly assume independence of sets in computing an intersection or a union, and if there are any terms common to both sets being intersected or unioned, independence no longer holds. While we cannot give a general proof of this heuristic, we can prove it separately for specific query forms. For instance,

SUP(SIZE(AND(A,OR(NOT(A),B))))
= SMALLEROF(SIZE(A),SIZE(OR(NOT(A)),B))
= SMALLEROF(SIZE(A),LARGEROF(SIZE(NOT(A)),SIZE(B)))
= SMALLEROF(SIZE(A),LARGEROF(SIZE(UNIVERSE)-SIZE(A),SIZE(B)))

whereas:

SUP(SIZE(AND(A,B)) = SMALLEROF(SIZE(A),SIZE(B))

and if SIZE(A) = 60, SIZE(B) = 35, and SIZE(UNIVERSE) = 100, the first form will give an upper bound of SMALLEROF(60,LARGEROF(40,35)) = 40, whereas the second form will give LARGEROF(60,35) = 35[12]. Since lower upper bounds are better, the second form gives a better answer in this case. Similar analysis can be applied to many other query-form pairs in regard to bounds rules, and some statistical analysis can be applied for estimate and error rules.

3.4.6.3. Distributive laws

The abovementioned absorption laws can be derived from more general distributive laws of intersection over union and vice versa, which are quite useful in their own right. That is, we should "factor out" parallel occurrences of the same symbol in expressions:

AND(OR(A,B),OR(A,C)) = OR(A,AND(B,C))
AND(OR(A,B),OR(C,A)) = OR(A,AND(B,C))
OR(AND(A,B),AND(A,C)) = AND(A,OR(B,C))
OR(AND(A,B),AND(C,A)) = AND(A,OR(B,C))

There are also distributive laws in regard to complements (alias DeMorgan's laws) which are useful in the following directions:

NOT(AND(A,B)) = OR(NOT(A),NOT(B))
NOT(OR(A,B)) = AND(NOT(A),NOT(B))

on the grounds of the heuristic that it's hard to apply complements to composite sets, but easier for simple sets.

Again, we can justify these in terms of bounds rules if necessary.

[12]SMALLEROF and LARGEROF are different from MIN and MAX: the former are binary operators taken in the standard mathematical sense, and the latter are statistical aggregate operations used in database queries on an indefinite number of objects.

3.4.6.4. Minimum-term form

It appears, then, that in nearly all cases we want to rewrite query sets in an equivalent minimum-term form. This form should be distinguished from (a) minimum-level form, and (b) minimum-operation form, both of which occur in the logic design literature [Givone 70]. Both of these have algorithms for finding them; the former is just simple disjunctive (or conjunctive) normal form, and the latter is approached via Karnaugh map theory. But minimum-term form has no nice algorithm for its discovery, except for a few special cases [Lawler 64], and the only way to finding it is exhaustive search. Thus we cannot give a general theory, only pairs of query forms where one form is superior to another.

3.4.6.5. Field-expression rearrangement

Field expressions can also be rearranged into equivalent forms. One can think of the names of real attributes of the database abstract as mathematical variables, and find the alternative forms with the standard laws of algebra. Just as with set-operation expressions, commutativity, associativity, and distributivity apply, and the minimum-term form is probably to be the best, following our heuristic. For instance,

PLUS(TIMES(F,G),TIMES(F,H)) should be replaced by TIMES(F,PLUS(G,H))

for purposes of estimate rules, even though it makes no difference to bounds rules. Squares, cubes, and other powers of attribute values are better than computing these things via TIMES since for instance

MEDIAN(Q,SQUARE(F)) = $(\text{MEDIAN}(Q,F))^2$

but:

```
SUP(MEDIAN(Q,TIMES(F,G))) =
  SMALLEROF(TIMES(MAX(Q,F),MEDIAN(Q,G)),
            TIMES(MEDIAN(Q,F),MAX(Q,G)))
INF(MEDIAN(Q,TIMES(F,G))) =
  LARGEROF(TIMES(MIN(Q,F),MEDIAN(Q,G)),
           TIMES(MEDIAN(Q,F),MIN(Q,G)))
```

hence:

```
SUP(MEDIAN(Q,TIMES(F,F))) =
  SMALLEROF(TIMES(MAX(Q,F),MEDIAN(Q,F)),
            TIMES(MEDIAN(Q,F),MAX(Q,F)))
INF(MEDIAN(Q,TIMES(F,F))) =
  LARGEROF(TIMES(MIN(Q,F),MEDIAN(Q,F)),
           TIMES(MEDIAN(Q,F),MIN(Q,F)))
```

and you get intervals, not exact values. Another example of the same thing has been given in section 3.3.1.

3.4.7. Miscellaneous rules

There are a few unusual rules that do not fall into any category thusfar discussed.

3.4.7.1. Database-domain-dependent rules

Our system is an expert system for the domain of statistical estimation, not the domain of the database. Our intention is to separate out this knowledge in the form of rules from database-dependent knowledge in the abstract. Thus we believe it important to write domain-dependent knowledge, as much as possible, as the value of a statistic, and not as an inference rule. For instance, the statement (as in [King 81]) that every ship nationality includes one freighter can be asserted for any particular database state by the fact that the statistic that the number of distinct nationalities for freighters is the same as the number of distinct nationalities for all ships in the database. The two assertions are not logically equivalent however -- the former concerns the intension of the database (things that are true for any database state), and the latter the extension (things that are true for only a particular database state).

Much of the so-called "domain knowledge" of [King 81] can be rewritten this way. For instance, any assertion about the intension of the database having the form

$$\forall x[P(x) \rightarrow Q(x)]$$

where P and Q are predicates corresponding to a classes of items in a relation, logically entails the assertion about an extension that the set corresponding to P is a subset of the set corresponding to Q, or in other words

$$\mathrm{SIZE(AND(PSET,NOT(QSET))} = 0$$

Similarly, any assertion about the intension of the database having the form

$$\exists x(P(x))$$

entails the assertion about the extension

$$\mathrm{INF(SIZE(PSET))} = 1$$

But unfortunately multiple quantifiers in the same expression like

$$\forall x \exists y[P(x) \rightarrow Q(y)]$$

cannot be captured this way.

3.4.7.2. Definitional-mode analysis

There is a fundamental interrelationship between intersections of pairs of simple sets and frequency statistics on both those sets, when simple sets are defined by partitions on the values of single attributes. For example, consider the set of American tankers, and assume we have statistics only on American ships and tankers separately. Consider the set of American ships, and its statistics with respect to the ship type field, and suppose that "tanker" is a ship type. The mode frequency for American ships is then an upper bound on the number of American tankers, and the frequency of the least most common item (the "antimode"), a lower bound. This is a powerful rule.

Some refinements need to be added for most situations, however, because usually the level of aggregation for first-order sets is coarser than the inherent aggregation of data values. For instance, only ship subtype may be entered, not type, whereas types may be the first-order classes. We must then find out how many different subtypes are in a given type, and, for instance, multiply the mode frequency by this number to get an upper bound.

3.4.7.3. Redundant intersections from range analysis

Sometimes it is beneficial to add redundant information to a query. An example is when a query set is known to span a certain narrow range with respect to some attribute, and you intersect a first-order set on that attribute containing that range with the query set. For instance, suppose that Egyptian ships never leave the Mediterranean, and that supertankers never venture west of Naples. Then we know that Egyptian supertankers (if there are any) are only to be found in the Eastern Mediterranean, and we may be able to exploit characteristics of that geographical population, as for instance using its size to upper-bound the number of Egyptian supertankers. Another example is item 8 of section 1.5.1.

Theoretically we can do this for any attribute, just appending additional set intersections. In practice we can do it only rarely, when the inferred range for some query set is fully contained in some first-order set defined on that range attribute. We can perhaps then rearrange the query using the rules of 3.4.6 and find even better forms, and so on.

3.5. The derivational dimension of rules

The last dimension of rules, the derivational one, differ from the others (statistic, answer, and computational) in that it does not have logical force, only heuristic. It represents the justification for a rule, what branch of knowledge it comes from. Since rules can often be supported by several lines of reasoning, a rule does not necessarily occupy a unique position on this dimension.

In section 3.5.1 we cover ways of deriving rules directly from mathematical laws and theorems. Section 3.5.2 covers some ideas from database theory. The next three sections concern three systematic formal ways of deriving query answers and rules: nonlinear optimization, maximum entropy theory, and theorem-proving manipulations of existing rules. Section 3.5.6 briefly mentions psychological rules people use for statistical reasoning.

3.5.1. Mathematics

Simple mathematical manipulations provide us with a large number of rules.

3.5.1.1. Definitions and theorems

We can directly take for rules many things expressed in mathematics and statistics books. For instance:

- a proportion statistic may be defined as the ratio of subset sizes to set sizes
- the variance (the standard deviation squared) is equal to the difference of the mean of the squares and the square of the mean
- the sampling theorem for means: the estimate (EST) of the mean of a subset is the mean of the set, and the error (ERR) of is σ * SQRT(1/N - 1/n), where N is the size of the subset, n the size of the set, and σ the standard deviation of the set
- the Chebyshev inequality for statistics (as opposed to probabilities): the number of items farther than distance D from the mean of a distribution can be no more than $n\sigma^2/D^2$, where σ is the standard deviation and n the total number of items
- the Cauchy-Schwartz-Buniakowski inequality: an upper bound on mean of the itemwise product of two attributes is the geometric mean of the arithmetic means of the squares of values on those two attributes

[Mitrinovic 64] gives some more simple inequalities of this sort.

3.5.1.2. Extreme cases of definitions and theorems

Analysis of definitions and theorems in particular cases can generate many useful rules. For instance, consider the formula for the mean, $\Sigma x_i/n$. Since $x_i \geq x_{max}$, $\Sigma x_i/n \geq nx_{max}/n = x_{max}$, and an upper bound on the mean of a set is the maximum, using the definitions of mean and maximum. Similarly, we know from the definition of median that we can renumber the x_i such that $x_i < x_{median}$ for $i \leq n/2$, and $x \geq x_{median}$ for $i > n/2$, for n even. Hence an upper bound on the *mean* of a set is

$$\Sigma_{i \leq n}[x_i/n] + \Sigma_{i > n/2}[x_i/n] \leq (n/2)x_{median}/n + (n/2)x_{max}/n$$
$$= (x_{median} + x_{max})/2$$

Another example is for the abovementioned Chebyshev inequality. If we suppose the maximum of some set to be a distance D from the mean, $1 \leq n\sigma^2/D^2$ hence $D \leq \sigma\sqrt{n}$, which says that a lower bound for any distribution on the distance between the maximum and mean is the standard deviation times the square root of the number of items.

3.5.1.3. Independence and linearity assumptions

A large class of results follow from the assumption that the data behave "reasonably" in one way or another. Exploratory data analysis [Tukey 77] in particular likes to make use of very simple data models, where data models are necessary. An important example is the "log-linear" model for data which represents set sizes or counts of intersections of two sets [Ku and Kullback 74]. It assumes that the sets are statistically independent of one another, and contribute to their intersection in a log-linear way:

$$\log(SIZE(AND(A,B))) = K_1\log(SIZE(A)) + K_2\log(SIZE(B)) + K_3$$

and the usual case is $K_1 = K_2 = 1$, $K_3 = \log(SIZE(UNIVERSE))$, or in other words

$$SIZE(AND(A,B)) = SIZE(A) * SIZE(B) / SIZE(UNIVERSE)$$

We can think of this as suggesting a two-dimensional contingency table where entries represent the number of items having pairs of attributes, and rows and columns must sum up to specified amounts. This sort of data model can be used for other statistics too. For instance:

$$MEAN(AND(A,B),F) = [MEAN(A,F) + MEAN(B,F)] / 2$$

where the mean is thought of as a logarithm of some fictitious quantity, and $K_1 = K_2 = 1$, $K_3 = \log(2)$. The same averaging formula can be used for standard deviation and median too. Actually, these computations are weighted in our system by the inverses of the relative sizes of A and B, so we do not use just a simple average, though the principle is the same.

The rules for our system that the above data models suggest are EST rules. That is, formally we should say

EST(SIZE(AND(A,B))) = SIZE(A) * SIZE(B) / SIZE(UNIVERSE)

because a data model is only an attempt to fit reality and not a depiction of it. ERR rules can also be derived from this analysis, as exemplified by the goodness-of-fit analysis of [Ku and Kullback 74].

3.5.1.4. Functional linearity

Another quite different way in which linearity assumptions lead to good rules is in the simplification of mathematically intractable expressions. A good example is the modelling of monotonic unary functional operations on numeric attributes, as when for instance we desire the mean of the logarithms of the values of some attribute. Unlike the median of the logarithms, there is no exact formula for this from the values of any simple statistics on the original attribute values. But since we know $\ln(x)$ is concave downwards because its second derivative $(-1/x^2)$ is always negative for $x>0$, any tangent to the curve $\ln(x)$ for $x>0$ will bound it from above; and any secant through two points of the curve will bound it from below in the interval between the two points.

Let either of these bounds lines be represented as $f(x) = ax + b$. Then if we wish a bound on $\ln(x_i)$ we use $ax_i + b$. Hence:

mean of $\ln(x_i)$ is $\Sigma \ln(x_i)/n$
which is bounded by $\Sigma[ax_i + b]/n = a\mu_x + b$

where μ_x is the mean of the original set of values x_i. The constants a and b are straightforward to obtain. If m is the minimum of the original set of values and M the maximum:

for the line tangent at $x=\mu_x$: $a = df(\mu_x)/dx = 1/\mu_x$, $b = \ln(\mu_x) - 1$
for the secant line: $a = [\ln(M)-\ln(m)]/[M-m]$, $b = \ln(m) - am$

So, substituting, we can say for the mean of the logarithms:

an upper bound is $\mu_x * 1/\mu_x + \ln(\mu_x) - 1 = \ln(\mu_x)$
a lower bound is $\mu_x[\ln(M)-\ln(m)]/[M-m] + \ln(m) - m[\ln(M)-\ln(m)]/[M-m]$
$= [\ln(M)-\ln(m)](\mu_x-m)/[M-m] + \ln(m)$
$= \alpha\ln(M) + (1-\alpha)\ln(m)$, $\alpha = (\mu_x-m)/(M-m)$

To give an example, if a set of data values ranges from 10 to 100, and the mean is 23, the mean of the logarithms of the data values has

an upper bound of $\ln(23) = 3.135$
a lower bound of $13/90 \ln(100) + 77/90 \ln(10) = 2.635$

We can do this for any monotonic function of one variable. The bounds on the mean of $f(x)$ for any monotonic $f(x)$ are

one bound: $f(\mu_x)$
other bound: $\alpha f(M) + (1-\alpha)f(m)$, $\alpha = (\mu_x-m)/(M-m)$

where μ_x is the mean of the x's, m the minimum on the range, and M the maximum. We can also bound standard deviation this way. And we can also get a tighter quadratic bound on the answer, but the mathematics becomes quite complicated.

3.5.1.5. Amounts of detail in rules

A key issue in using mathematics to get rules is the amount of detail one is willing to put up with. Frequently one can get a somewhat better (tighter) bounds rule by adding extra clauses and terms, while requiring more space and processor time to evaluate, and running greater risk of errors. One must decide if the tradeoff is worth it. For instance, a lower bound on the number of distinct items in a set is the size of the set divided by its mode frequency. But if one also has a reasonbly tight value for the second most common item in the set (MODEFREQ2, a value which we do tabulate routinely for database-abstract sets), a better (larger) lower bound is

$$1 + [[\text{SIZE}(A) - \text{MODEFREQ}(A)] / \text{MODEFREQ2}(A)]$$

And we can prove this rule is better:

$$\begin{aligned}\text{SIZE}(A) / \text{MODEFREQ}(A) \\ &= 1 + [[\text{SIZE}(A) - \text{MODEFREQ}(A)] / \text{MODEFREQ}(A)] \\ &\leq 1 + [[\text{SIZE}(A) - \text{MODEFREQ}(A)] / \text{MODEFREQ2}(A)]\end{aligned}$$

because $\text{MODEFREQ}(A) \geq \text{MODEFREQ2}(A)$.

3.5.2. Database theory

Database theory provides a different way of obtaining rules.

3.5.2.1. Keys and functional dependencies

Keys are attributes or "fields" in database terminology (or groups of fields, through vectorization) of a relation whose values together identify a unique tuple [Ullman 80]. In this work we broaden the concept to include "extensional keys", fields whose unique-identification property holds only in the current database state, as we mentioned in section 3.1. Such information is quite useful because the number of distinct values for sets which restrict values for all or some of the key fields can have much better bounds than otherwise.

Extensional functional dependencies are a related more general concept. There is a functional dependency for field A to field B if knowing the value of an item for field A logically implies its values for field B. This is useful because, for instance, it puts an upper bound on the number of distinct items for field B for some set as the number of distinct items for field A for the same set.

We do not use functional dependencies in our implementation, however, since they are a kind of correlation (see section 3.1.3).

3.5.2.2. Theory of inference compromise

Frequently statistical databases must satisfy the two conflicting goals of giving out values of statistics and protecting the confidentiality of individual record (tuple) values. Research has explored conditions for violating this confidentiality when giving out statistics, the conditions for "compromise" [Fernandez et al 81, Denning 83]. While the aim of our system is not inference of individual values, the same mechanisms can often be used for the (less difficult) problem of inference of statistics on arbitrary sets. Four important compromise methods are useful for our purposes:

- "trackers" [Denning et al 79], formulas that represent a "padding" of another query so that the latter can be "subtracted" from the former to get statistics on the difference set. For example, to find the number of Israeli ships with atomic weapons, subtract the number of non-Israeli ships with atomic weapons from the number of ships with atomic weapons, both the latter of which may be large. We use this ideas in many guises in our system.

- small-set inferences, the unique or near-unique solution of systems of constraints on very small sets. For instance, if one knows a set is no larger than four items, exact values for the maximum, minimum, mean, and median -- or even many kinds of bounds on them -- exactly determine what those values are. The U.S. Census Bureau does analysis of this kind of compromise on all statistical summary tables before they release them to the public [Cox 80]. A number of our rules have this effect when set sizes are small, so we did not implement rules explicitly.

- key inferences, exploitation of the fact that fields (or the vectorizations of fields) are extensional keys of a relation. For instance, if we know that ship ID number is an extensional key of a ships relation, that the maximum ID number of ships involved in accidents last year was 2694410, and that the median ID number of American ships last year was also 2694410, then the American ship with that ID must have been involved in an accident. We did not implement this kind of inference because it requires coincidences which will occur rarely in large databases.

- Diophantine compromise. The additional constraint on a set of variables that they must be integers (the "Diophantine" constraint) is usually a very powerful one. Any situation where counts are unknown is an example, but in fact any unknowns that are rational numbers can always be scaled upwards, turning the problem in a Diophantine one. Standard algorithms exist for solving sets of linear Diophantine equations, though the size of the solution space for a problem may vary tremendously. We discuss this inference method in detail in [Rowe 83b]. We have done a number of experiments with an implementation of this method, but we have not integrated it into the full system because solving linear Diophantine equations is an NP-complete problem, requiring exponential amounts of time in the number of variables, and hence requires care before application.

3.5.3. Nonlinear optimization

Finding out absolute bounds on functions also goes by the name of "optimization", and has been subject of extensive attention in operations research [Gill, Murray, and Wright 81], some of which is relevant to us. (We have not, however, used these methods much in our implementation, for reasons discussed in section 3.5.3.2.)

3.5.3.1. The optimization process

The general optimization problem is to find the maximum of some function f of n variables x_i. (This corresponds to an upper bound; the lower bound is found by optimizing -f.) In the case of statistics, f is some statistic like maximum, mean, standard deviation, median, etc. and the x_i are data points. Note right away one problem: one must know exactly how many data points there are in advance for this to work, i.e. one must have an exact value for the SIZE statistic on the set. Then one gets additional information in the form of "constraints" on the solution, which for our statistics domain will be values of other statistics on the same or overlapping data points. Some of these constraints can be equality constraints, e.g. mean or standard deviation; some can be inequality constraints, e.g. maximum, median, or maximum of a distribution fit; and some can be neither, as the frequency of the mode. So a typical situation might be to find the largest value of the mean of a set, given values for the maximum, minimum, median, and number of distinct values.

Note that uncertainty in the values of known statistics is straightforward to handle if we know an absolute range. For instance, if we don't know the exact value of the mean but know it is in the range 12 to 17 we can write two inequality constraints instead of the one equality constraint we would have if the value was known exactly, namely

$$\Sigma x_i/n \geq 12, \Sigma x_i/n \leq 17$$

Once the problem is set up, optimization techniques attempt to characterize the behavior of the function f in the range given by the constraints. This usually involves calculating its gradient and following it (e.g. steepest ascent method), perhaps also finding the Hessian (second-derivative) matrix and using it to control step size. A good starting point for search for the optimum is often important.

3.5.3.2. Disadvantages of optimization

There are three difficulties with using optimization in our system, however. First, it only works for specific cases, not in general. That is, we cannot get general rules from it that characterize the solution for a class of known values, only answers for specific known values. There are some other techniques that will do this, but they only work for a few cases -- see section 3.5.3.3.

The second problem is that the optimum found this way is a local optimum, not necessarily a global one. Few methods have been found for that latter kind of problem. But if one does have some knowledge of the behavior of the function being optimized in the form of its gradient and Hessian, one may be able to prove globality in special cases. For instance, when the objective function is linear.

The third problem is that in most interesting cases there are many variables, since the variables represent data points and their number is the size of the set being studied. Since in most cases they are interchangeable, this is an inefficient paradigm for determining things about them.

3.5.3.3. Some tractable special cases

An important special case where we can get "nice" general solution is when constraints are linear and the objective (optimization) function is linear as well. An example would be when we know set size (assume it even), maximum, minimum, and median (call these n, M, m, and d respectively) and we wish to know the largest possible value of the mean. Then if we assign indices to the variables in order, i.e. $x_1 \leq x_2 \leq ... \leq x_{n-1} \leq x_n$, we can write the constraints

$$x_1 = m$$
$$x_n = M$$
$$x_{n/2} = d$$
$$m \leq x_i \leq d \text{ for } 1 \leq i \leq n/2$$
$$d \leq x_i \leq M \text{ for } n/2 \leq i \leq n$$

and the objective function is

$$x_1/n + x_2/n + ... + x_{n-1}/n + x_n/n$$

So the gradient vector is

$$(\partial f/x_1\ \partial f/x_2\ \partial f/x_3 \ldots.) = (1/n\ 1/n\ 1/n \ldots.)$$

and is constant in the feasible region (constant everywhere, in fact). So we can use the Simplex method in the usual linear programming way to show the maximum occurs for

$$x_1 = m$$
$$x_i = d, 1 < i \leq n/2$$
$$x_i = M, d < i \leq n$$

and hence an upper bound on the mean is

$m/n + (.5 - 1/n)d + .5M$

If, say, we have the same constraints and wish to bound the standard deviation, we have a quadratic programming problem which also has some nice properties, albeit not quite as nice. But if a standard deviation is one of the constraints, things are considerably messier.

3.5.4. Maximum-entropy theory

Symbolic optimization provides a nice metaphor for how one goes about deriving bounds rules. Can we do anything similar for estimate and error rules? It turns out that we can, if we restate the problem as one of finding the "maximally nice" probability distribution consistent with some given information. For "nice", there are many good reasons to take it as meaning "maximum-entropy" [Shore and Johnson 81]. The entropy associated with an arbitrary probability distribution p(x) may be quantified as:

$$-\int p(x)\ln(p(x))\,dx$$

For instance, the entropy of the even distribution $p_1(x) = 1$ on the range $0 \leq x \leq 1$ is

$$-\int 1 * \ln(0)\,dx = 0$$

whereas the entropy of the exponential distribution $p_2(x) = e^{-x}/1-e^{-1}$ on the same interval is

$$-\int e^{-x}/1-e^{-1} * x\,dx = -[e^{-x}(-x-1)/(-1)^2]_0^1 / 1-e^{-1}$$
$$= [2e^{-1} + -e^0]/1-e^{-1} = -.4180$$

so $p_1(x)$ has a larger entropy than $p_2(x)$, and so is more "natural".

3.5.4.1. Finding the maximum-entropy distribution

Our problem then becomes one of finding a *distribution* that maximizes a single number, the entropy, given certain prior information. This is now a problem for the calculus of variations [Korn and Korn 68]. For this particular form of function to be evaluated it turns out the solution always has the general form:

$$f(x) = e^{\lambda_0 + \lambda_1 x + \lambda_2 x^2 + \ldots}$$

where the λ_i are constants, the so-called "Lagrange multipliers".

The solution takes simple forms in important cases.

- If nothing whatsoever is known about a distribution, then the maximum-entropy distribution has only the first λ term, and is perfectly even.
- If only the mean of a distribution is known, only the first two (λ_0 and λ_1 terms) occur, and the maximum-entropy distribution is an exponential.

- If the mean and standard deviation are known, then only the first three terms occur. It can be proven that necessarily $\lambda_2 < 0$, giving a displaced normal curve. (This in fact is a "deep" justification of the Central Limit Theorem of probability.)

- If a maximum and/or a minimum is known in addition to a mean and/or standard deviation, the maximum-entropy curve is just a truncated version of the curve without the maximum and/or minimum.

- If medians or other order statistics are known, the curve is piecewise-constant, piecewise-exponential, piecewise-normal, etc.

- If the number of distinct values in a set as well as the set size is known, the mean number of items per value is their ratio. Thus the maximum-entropy curve for a *frequency* distribution of values where the horizontal axis is frequency of occurrence and the vertical axis is number of items having that frequency of occurrence is an exponential. The curve is bounded on the high end by the mode frequency and on the low end by the frequency of the least common item if these are known, otherwise by the size of the set on the high end and zero on the low end. Complicated arguments can be used to relate this to the Yule distribution, a distribution very close to the simple reciprocal predicted by Zipf's "Law" [Rapopport 78].

3.5.4.2. Using the maximum-entropy distribution

For getting ESTs and ERRs for our system, then, we just compute respectively the mean and standard deviation of the maximum-entropy inferred distribution. Unfortunately, the mathematics of finding the Lagrange multipliers cannot be done in closed form in all but a few cases, and numerical methods must be used, analysis of whose accuracy is sometimes difficult; [Shore and Johnson 81] discusses this in detail. Again, it is difficult to generalize these answers, as results of a kind of optimization, into *rules* for classes of answers.

3.5.5. New rules from old

A very important way of obtaining rules, perhaps *the* most important, is to work from existing rules in various ways.

3.5.5.1. Rule composition

Rules can be created from the functional composition of existing rules. This is particularly important in the defining of new statistics from existing ones. A simple example is defining the "proportion" statistic as the ratio of the set-size statistic of some set to the size of the universe. Then the rule

SUP(SIZE(AND(A,B))) = LARGEROF(SIZE(A),SIZE(B))

transforms into

SUP(PROPORTION(AND(A,B))) = SUP(SIZE(AND(A,B)) / UNIVERSESIZE)
= SUP(SIZE(AND(A,B))) / UNIVERSESIZE
= LARGEROF(SIZE(A),SIZE(B)) / UNIVERSESIZE
= LARGEROF(PROPORTION(A),PROPORTION(B))

An important application of functional composition is to the mean of the pairwise product of two numeric attributes, which is related to the linear correlation of those two attributes through the relationship

$$\mu_{X*Y} = \mu_X\mu_Y + \rho_{XY}\sigma_X\sigma_Y$$

where μ's are means, σ's standard deviations, and ρ_{XY} a "correlation coefficient" between X and Y that ranges from 1 (perfect correlation) through 0 (no correlation) to -1 (perfect negative correlation). We can get results such as:

SUP(MEAN(Q,TIMES(F,G))) =
[MEAN(Q,F) * MEAN(Q,G)] + [SIGMA(Q,F) * SIGMA(Q,G)]
INF(MEAN(Q,TIMES(F,G))) =
[MEAN(Q,F) * MEAN(Q,G)] - [SIGMA(Q,F) * SIGMA(Q,G)]

A different example is derivation of the rules for handling subsets in intersections and unions from the rules for handling disjoint sets, using the fact that set A is a subset of set B if and only if the complement of B is disjoint from A. Then the rule

SIZE(OR(A,B)) = SIZE(A) + SIZE(B) if A and B are disjoint

maps into

SIZE(OR(A,NOT(B))) = SIZE(A) + SIZE(NOT(B)) if A is a subset of B

3.5.5.2. Inversion and rearrangement of rules

As we mentioned in 3.4.2, rules (especially statistic-statistic ones) can be rearranged in various ways, and the best form usually ought to be chosen in advance. Bounds rules are equivalent to inequalities, and different single terms can be moved to either side of the inequality to get different forms. For instance

SUP(MODEFREQ(OR(A,B))) = MODEFREQ(A) + MODEFREQ(B)

really means

MODEFREQ(OR(A,B)) $\leq$ MODEFREQ(A) + MODEFREQ(B)

Estimate and error rules can be thought equivalent to equalities, with however the provision that substitution of equivalent forms is not allowed, i.e. just because

EST(SIGMA(A)) = [MAX(A) - MIN(A)] / SQRT(12)
EST(SIGMA(AND(A,B))) = [SIZE(B)SIGMA(A) + SIZE(A)SIGMA(B)]
/ [SIZE(A) + SIZE(B)]

does not mean

[MAX(AND(A,B)) - MIN(AND(A,B))] / SQRT(12)
= [SIZE(B)SIGMA(A) + SIZE(A)SIGMA(B)] / [SIZE(A) + SIZE(B)]

Note that one must be careful about signs in manipulating inequalities, and this may prohibit certain forms of inequalities. For instance, if

MEAN(Q,F) * SIZE(Q) $\leq$ MEAN(UNIVERSE,F) * SIZE(UNIVERSE)

that means that

MEAN(Q,F) $\leq$ MEAN(UNIVERSE,F) * SIZE(UNIVERSE) / SIZE(Q)

(provided SIZE(Q) is nonzero) but not that

SIZE(Q) $\leq$ MEAN(UNIVERSE,F) * SIZE(UNIVERSE) / MEAN(Q,F)

unless MEAN cannot be negative.

3.5.5.3. General theorem-proving

Certain kinds of interesting rules cannot be derived from the above "shallow" analyses, but require going back to the original definitions of statistics and reasoning from there. Standard artificial-intelligence theorem provers can used for doing this automatically. A simple example is putting bounds on the median given the maximum and minimum. Since

"M is the maximum of set A" $\leftrightarrow \forall x \in A(x \leq M)$
"m is the minimum of set A" $\leftrightarrow \forall x \in A(x \geq m)$

and since one of the conditions on the median is

"d is the median of set A" $\rightarrow d \in A$

we logically conclude that for the particular $x = d$, $m \leq d \leq M$.

As another example, consider the following way of deriving both upper and lower bounds on the mode frequency given the size of a set (n) and the number of distinct items in the set (m). Define the frequencies of values of the set as $f_1, f_2, \ldots, f_m$, all nonzero. Then:

$$n = f_1 + f_2 + \ldots + f_m$$

By the definition of the mode frequency r of a set,

$$\forall i[r \geq f_i]$$

hence substituting in the first equation

$$n \leq r + r + \ldots + r = mr$$

hence $r \geq n/m$, i.e. the mode frequency is bounded below by the ratio of the set size to the number of distinct values in the set. We can use analogous reasoning to get an upper bound. Since by the definition of the f_i

$$\forall i[f_i \geq 1] \wedge \exists i[f_i = r]$$

therefore

$n \geq r + 1 + 1 + ... + 1 = r + (m-1)$

and $r \leq n + 1 - m$, i.e. the mode frequency is bounded above by the size of the set plus 1 minus the number of distinct values.

3.5.5.4. Exploiting analogies between rules

Rules tend to show symmetries, and in many cases, form "families" (which may even be groups in the group-theoretic sense). Thus an important way of getting new rules is to derive one rule, analogize a corresponding rule, and then verify its correctness. The last step is particularly important, since there are often misleading symmetries and surprising special cases. But since theorem-proving is considerably easier than theorem-finding (a good algorithm, namely resolution, exists for proving a large class of theorems), good analogies can be very useful.

One important general symmetry can be suggested, however: the symmetry of rules for maxima with rules for minima (not to be confused with the SUP and INF bounds). The two are always related by the following:

MIN(A,F) = -(MAX(A,MINUS(F)))

i.e. the maximum of a set with respect to an attribute is the negative of the minimum with respect to the negative of that attribute. We thus can rewrite the rule

MAX(OR(A,B),F) = LARGEROF(MAX(A,F),MAX(B,F))

as

```
MIN(OR(A,B),F) = -(MAX(OR(A,B),MINUS(F)))
               = -(LARGEROF(MAX(A,MINUS(F)),MAX(B,MINUS(F))))
               = -(LARGEROF(-MIN(A,F),-MIN(B,F)))
               = SMALLEROF(MIN(A,F),MIN(B,F))
```

and we have the corresponding rule for MIN. (We assume laws for manipulating LARGEROF, SMALLEROF, and -.) Something similar can be done for other MAX rules.

As an example of a family of rules, consider the following regarding maxima and minima of sums and differences of corresponding attribute values for sets:

SUP(MAX(Q,PLUS(F,G)) = MAX(Q,F) + MAX(Q,G)
INF(MAX(Q,PLUS(F,G)) = LARGEROF(MAX(Q,F) + MIN(Q,G),
MIN(Q,F) + MAX(Q,G))
SUP(MIN(Q,PLUS(F,G)) = SMALLEROF(MIN(Q,F) + MAX(Q,G),
MAX(Q,F) + MIN(Q,G))
INF(MIN(Q,PLUS(F,G)) = MIN(Q,F) + MIN(Q,G)
SUP(MAX(Q,DIFFERENCE(F,G)) = MAX(Q,F) - MIN(Q,G)
INF(MAX(Q,DIFFERENCE(F,G)) = LARGEROF(MAX(Q,F) - MAX(Q,G),
MIN(Q,F) - MIN(Q,G))
SUP(MIN(Q,DIFFERENCE(F,G)) = SMALLEROF(MIN(Q,F) - MIN(Q,G),
MAX(Q,F) - MAX(Q,G))
INF(MIN(Q,DIFFERENCE(F,G)) = MIN(Q,F) - MAX(Q,G)

An example where symmetries are not exact but more complex occurs with the eight cases involving bounds on MAXGAP and MINGAP (largest distance between successive different ordered values) of intersections and unions:

SUP(MAXGAP(AND(A,B))) = ∞
INF(MAXGAP(AND(A,B))) = LARGEROF(MAXGAP(A),MAXGAP(B))
SUP(MINGAP(AND(A,B))) = ∞
INF(MINGAP(AND(A,B))) = LARGEROF(MINGAP(A),MINGAP(B))
SUP(MAXGAP(OR(A,B))) =
LARGEROF(MAXGAP(A),MAXGAP(B),DISTBETWEEN(A,B))
INF(MAXGAP(OR(A,B))) = 0
SUP(MINGAP(OR(A,B))) =
SMALLEROF(MINGAP(A),MINGAP(B),DISTBETWEEN(A,B))
INF(MINGAP(OR(A,B))) = 0

where

DISTBETWEEN(A,B) = 0 if A is not disjoint from B on the range
= LARGEROF(MIN(A)-MAX(B), MIN(B)-MAX(A)) otherwise

Clearly there are symmetries among the eight rules where MAXGAP is interchanged with MINGAP, SMALLEROF with LARGEROF, AND with OR, and SUP with INF, but the addition of the DISTBETWEEN term is a surprise, and also its nonoccurrence in the sixth and eighth rules. These symmetries allow us to find rules by finding a few of a class, then postulating others by analogy.

(Note there are other rules for these same situations -- these are only the set-decomposition rules. Note in particular some important statistic-statistic rules (see 3.4.2 can be used to get better bounds in rules 1, 3, 6, and 8 above:

SUP(MAXGAP(A)) = MAX(A) - MIN(A)
SUP(MINGAP(A)) = MAXGAP(A)
INF(MAXGAP(A)) = MINGAP(A)
INF(MAXGAP(A)) = [MAX(A) - MIN(A)] / SIZEUNIQUE(A)
SUP(MINGAP(A)) = [MAX(A) - MIN(A)] / SIZEUNIQUE(A)

where SIZEUNIQUE is the number of distinct items in a set, which have their own symmetries.)

3.6. Two rule categories not implemented

Two important rule categories of the taxonomy of chapter 1 and appendix C were not implemented, for important reasons: join rules and prototype rules.

3.6.1. Join decomposition

In addition to specifying a set of items and a set of their attributes of interest, queries must specify a database relation as we are assuming a relational database model. Since we have covered the relational operators of restriction, projection, union, intersection, and complement by our previous discussion, all that remains for relational completeness [Ullman 80] is handling of joins between relations (or alternatively, the inverse of the join operation, sometimes called the "quotient", but we shall not discuss it).

A few general comments are in order. If we expect a join to be frequently requested, we may tabulate statistics on its attributes in advance, just as with any other relation. (We often know in advance that just a few joins are semantically meaningful, from our data model or schema.) But otherwise we may only have statistics on the individual relations being joined. In this case, the join may be seen as a weighting of the values of one relation by distributional characteristics of the other. If the join column(s) form an (extensional) key for the second relation, the weights are constant, and the new relation is the same as the old except for some new added columns. Otherwise, to get estimates one must determine the maximum and minimum weightings for each term, and how they would change the values of statistics computed on particular fields of interest. The bounds on such new statistics are often weak because the extra uncertainty associated with the join multiplies the uncertainty in calculating the statistic on the original relation.

Thus, except for some nice cases, joins are not well handled by our system (though we can take comfort in the fact that all relational systems find joins a problem in one way or another). We have not implemented join handling in our demonstration system for this reason, and also because both test databases were naturally expressed as single relations.

3.6.2. Reasoning with prototypes

One other category of rules includes those that people apparently make judgments of statistics on sets. In general, people use a simple method subject to numerous fallacies [Rosch and Mervis 75, Rips 75, Kahneman and Tversky 82]. This behavior seems to be well modelled for most domains by metric-distance models of points in hyperspace. That is, first-order sets are representing by locations in an n-dimensional Cartesian coordinate system, and some queried set is also represented by a location; lateral inheritance (see section 3.4.5) then estimates statistics on the query set from some sort of average of statistics on "nearby" points in the coordinate system, their importance being inversely weighted by their distance from the query set location. As an example, consider figure 3-2. The dimensions represent two properties of ships, tonnage and variety of goods carried; and five sets are represented, American ships, Liberian ships, tankers, freighters, and a queried set of American tankers about which no statistics are known. So "the average American tanker" might be three times closer to the set of American ships than to the set of tankers, and hence its length might be 3/4 times the average length of an American ship plus 1/4 times the average length of a tanker.

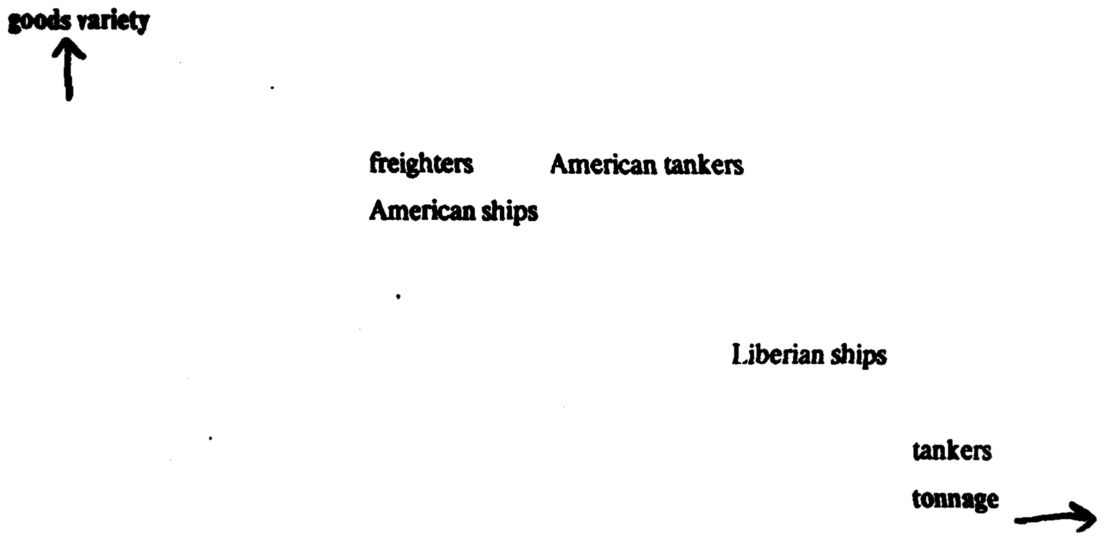


Figure 3-2: Example of statistical estimation from metric-space prototypes

Nonnumeric attributes can also be handled by this method -- for instance, "Hamming distance" can be used which is the number of attributes in common between two entities specified as property lists -- but much effort has gone into finding numeric dimensions for even very clearly nonnumeric data.

Reasoning with prototypes is a highly database-dependent method -- dimensions, scales, and point locations must be found for each attribute -- and is thus highly incompatible with the database-independent character of our statistical inference system. Users may not be consistent in their behavior with such models, and they may be missing particular prototypes that others have, or have additional prototypes. Also, since it is a form of lateral inheritance, the reasoning does not give bounds on statistics, which we consider a serious weakness since many applications are only interested in bounds. For these reasons we do not use it in our system.

3.7. Global issues for rules

There are a few general issues regarding the rules which do not fit anywhere in the preceding discussion.

3.7.1. *Completeness of the rules*

We have a large rule-based system (about 400 rules in the current implementation), and it is important to use the rule taxonomy during design to look for gaps in coverage. But general statistic-statistic rules can always be used to give weak bounds on answers if no strong rule can be thought of. This does happen, in the same way that in a symbolic algebra system there are functions that have no closed form integral. In fact, it may be possible to *prove* as in symbolic algebra that in certain rule situations that no rules of a certain form could ever exist. Inability to find a closed-form maximum-entropy estimator for AND(A,AND(B,C)) given the sizes of AND(A,B), AND(A,C), and AND(B,C) (cf. [Ku and Kullback 68]) may just represent such a fundamental limitation. Note there *is* an estimator given only sizes of A, B, and C, namely

SIZE(A)*SIZE(B)*SIZE(C)/SIZE(UNIVERSE)*SIZE(UNIVERSE)

and there is one given only sizes of B, AND(A,B), and AND(B,C):

SIZE(AND(A,B))*SIZE(AND(B,C))/SIZE(B)

so some superficially similar situations are easy while others are hard.

Observations like these suggest a general principle is justifying our rule-based approach: when a priori knowledge situations become too complicated, closed-form rules are not possible, only computationally expensive global methods like maximum-entropy estimation [Cheeseman 83]. Our goal is to "partition" complex situations into pieces small enough so that maximum-entropy and nonlinear-optimization methods can be precompiled where a closed form solution exists. Unfortunately, this partitioning does not ensure a set of maximum-entropy solutions on the pieces is a maximum-entropy solution for the whole situation, but it is a good heuristic.

3.7.2. Degree of detail of the rules

As we noted in 3.5.1.5, rules can be formulated to varying degrees of precision, depending on how critical time, space, and accuracy are for certain classes of queries. (In our implementation we have generally tried to keep rules simple, except for a few important but complicated ones.)

3.7.3. Redundancy of the rules

Rules that are subsumed by other rules can often be suggested experimentally. One just keeps track of which rules "contributed" to answers in a large corpus of queries, which means that the final answer would change if that rule were omitted. The easy case for determining this is when the value resulting from a rule appears in the corresponding place (bounds, estimate, or error) in the final answer quadruple, and no binary operations were used in getting the answer. Binary operations require a table of backwards-propagation dependencies derived from our quadruple algebra definitions, which for instance for DIFFERENCE looks like this:

- if SUP is in error in a difference, examine the SUP of the first argument and the INF of the second
- if INF is in error, examine the INF of the first argument and the SUP of the second
- if EST in error, examine the EST of the first argument and the EST of the second
- if ERR is in error, examine the ERR of the first argument and the ERR of the second

Redundancy can be never proved by finite experimentation, however, only suggested; theorem-proving methods are needed for proof.

3.7.4. Debugging the rules

Debugging can use the same "blame assignment" methods as redundancy-checking. Unlike many other expert systems, a clear and unambiguous right answer exists for all problems our system attempts to solve, namely the answer to the same query on the full database from which the database abstract is abstracted. If an estimate strongly disagrees with the answer, all the rules that contributed to it are under suspicion, and the predictions of each may be individually compared to the values on the full database.

Consistency between the answer and the bounds provides only one kind of debugging, however. We can also compare bounds on queries on related sets for expected narrowings consistent with the inheritances of section 3.4.5. For instance, the SUP on the mode frequency of a subset must not exceed the SUP of the mode frequency on the set, for the same attribute, or else there is a bug in the bounds rules.

3.7.5. Storage of the rules

Compared to the size of database abstracts we envision, rule storage is negligible. Rule action parts are mostly simple, using just a few arithmetic operations, codable in just a few bits each. Rule condition parts can be coded using the first three dimensions of the taxonomy (statistic, characteristic, and computational), and the code can be explained in a separate table. Since we have 400 rules in our current implementation, and we envision on the order of a thousand in a reasonably complete system, this is not a major storage issue.

Chapter 4
Methods: the database abstract

Since the discovery of the Pervertimento, even his detractors have had to admit that P. D. Q. Bach must be considered history's greatest late-eighteenth-century Southern German composer of multi-movement works for bagpipes and chamber orchestra.

Peter Schickele, The Definitive Biography of P. D. Q. Bach, *Random House, 1976*

We now shift attention to the other half of our system, the database abstract. Section 4.2 addresses design criteria for the abstract, and section 4.3 explains the physical storage scheme. Section 4.4 discusses management of construction of the abstract, and section 4.5 keeping it consistent with the database.

4.1. The abstract as closed world

The expertise of this expert system is in the domain of statistics, not the domain represented by a particular database as in [King 81]. The database abstract itself thus represents *all* the knowledge of the database domain that system has available, and it is the job of a database administrator to guarantee that its contents maintain "integrity" with regard to the real world, representing the world accurately and completely.

4.2. Choosing what to put in the abstract

System performance is quite sensitive to the contents of the database abstract. Different databases abstracts of the same size can have markedly different estimation behavior. It is thus important to carefully plan out the contents of the database abstract.

Unfortunately, however, this is just as much an art as a science. Frequency statistics for query types may be very difficult to estimate if the database is new, and even if the database has been queried for a long time, the very different query responses and response time for the database abstract and rules may evoke new querying patterns. Also, our rules interact in many complicated ways -- this is a primary *justification* -- and including one item of information in the database abstract may make seemingly unrelated information mostly redundant. Thus the most frequently queried sets should not necessarily be the ones loaded into the abstract: only some of them may give nearly the same effect. Selecting the best sets to put into a fixed-size database abstract is therefore a difficult problem in optimization of a nonlinear function of a large number of variables, probably much too complicated a problem to solve completely in reasonable time. Guidance from an expert in the domain of the data could be helpful. Perhaps we can think of our system as a "double expert system" here: a statistics expert provides the rules, and an expert in the domain of the database constructs the abstract.

A few guidelines for what to put in the abstract are possible, though, based on intuitive arguments.

4.2.1. The universe

There are sets on which it is essential to have a quite complete set of statistics for *any* database: the sets which correspond to complete relations of the database (or "universe" sets). Since downwards inheritance can always give bounds on statistics, it is essential there be a "last resort" set from which weak absolute bounds can be inferred when nothing else is known. For another thing, the quadruple algebra has trouble with null bounds, and weak bounds are much preferable to none.

4.2.2. First-order sets

For building database abstracts we have found it useful to exploit the concept of a "first-order set" (see 1.8.1). These are sets defined as disjoint partitions of the values of a relation as per values of a single attribute of that relation. For instance, "tankers" is a partition of ships on the ship-type attribute; "American ships" a partition on the nationality attribute; "ships 300 to 400 feet long" on the length attribute. We believe that such sets are often the conceptual building blocks which people use to describe a domain, and thus they should be the building blocks of a database abstract. More complicated sets can be described as subsets, complements, intersections, and unions of them. E.g., "American tankers" is the intersection of the set of American ships with the set of tankers, and "North American ships" is the union of United States ships and Canadian ships.

As an example, 18 first-order sets were defined for the experiments on the medical database discussed in chapter 6, as follows:

- patient-number attribute: visits of patient 3, visits of patient 6, visits of patient 10
- sex attribute: male patient visits, female patient visits
- disease-activity attribute: visits where patient was not sick, visits where they were mildly sick, visits where they were moderately sick, visits where they were very sick
- temperature attribute: visits where the patient temperature was below normal (37.5 centigrade) , visits where temperature was normal, visits where temperature was above normal
- cholesterol attribute: visits where the patient had low (< 230 units) cholesterol, visits where cholesterol was medium (230 units), visits where cholesterol was high (> 230)
- prednisone attribute: visits where patient was being dosed with low amounts of prednisone ($\leq$ 10 units), visits with medium prednisone (11 to 24), visits with high prednisone ($\geq$ 25)

Disjointness of the first-order sets defined on the same attribute is very important, because the inference rules for unions (and intersections too, naturally) of disjoint sets are generally exact instead of approximate, and exact rules for statistics are highly valuable. For instance, disjoint unions are exact for set sizes, means, maxima, minima, standard deviations, and some of the time for modes, mode frequencies, and medians too. Nondisjoint unions are only exact for maximum and minimum.

4.2.3. Selection of the attribute partitioning

There are three very different, often contradictory objectives to be addressed in defining the first-order sets: semantic meaningfulness, evenness of coverage of user queries, and evenness of coverage of the data. Since users tend to ask queries about semantically meaningful sets of items, it may be a good idea to try to develop first-order sets from classical psychological "clustering" results on domains, e.g. [Rosch et al 76] which suggests there is some optimal "middle" level of category description that people tend to prefer in thinking and talking about common natural phenomena, like animals and furniture. However, databases are not very "natural" for the most part since many reflect artificial environments and data not primarily intended for human consumption, and thus it may be foolish to apply psychological clustering methods blindly to all attributes. But it may work for attributes that have well-defined common-consent natural-language categories.

On the other hand, database research suggests that best performance occurs when database access paths are adjusted for even usage frequencies, i.e. even "loading" ([Wiederhold 77], section

5-4-4). An example is trying to maximize the average fraction of user queries that can be answered directly from the abstract. So if users are asking three times as many questions about tankers as freighters, we should define three times as many subtypes of tankers for first-order sets as subtypes of freighters. Utilities of different kinds of queries may be introduced as a way of weighting query-form frequencies by importance. The difficulty with access-frequency optimization is that it often does not address fundamental structural differences in a domain that can lead to significant, exploitable statistical differences between classes.

A third option is to make the first-order sets denote equal-sized partitions of their attributes, that is, equal in the number of database items that correspond to them. Of course, this can only work for a particular database state, and if the database contents change (e.g. many inserts occur of items belonging to a particular first-order class), evenness may no longer be assured, and it is costly to construct a whole new database abstract. The advantage, however, is that a relatively uniform level of accuracy can be provided for query answers on all sets of the same size. For instance for means, as we mentioned in section 3.4.5, the error in an estimate of the mean of a subset of some set with known mean is $\sigma\sqrt{(1/N - 1/n)}$ where n is the size of the set with the known mean, σ its standard deviation, and N the size of the subset; the smaller the subset, the worse this estimate is. Something similar occurs for all other statistics.

These three approaches are all important to varying degrees in different database query situations. How they affect the tradeoff between space, time, and accuracy in a system is complex and more a matter of judgment than formal analysis.

4.2.4. Grain of the attribute partitioning

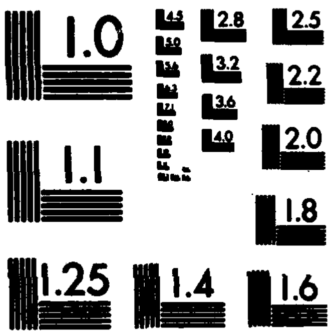
A related issue is that of how many first-order sets one should have for a given database. If the database abstract consists only of first-order sets, then obviously more sets means a bigger abstract and better estimation accuracy (though probably not an increase in calculation time -- see discussion in 6.6). If one also allows statistics on "second-order sets" (intersections and unions of two first-order sets) and sets representing non-disjoint partitions of attribute values, the abstract may become much larger. Intuitive judgment may be the best guide. It may be a good policy, however, to err on the side of too many first-order sets in design, since inference upwards on disjoint sets is more robust and demonstrates less sensitivity than inference downwards to subsets.

Some formal methods do apply to choosing the grain size in the case of an entirely first-order abstract. Consider the first-order sets that represent even partitions on the values of an evenly

AD-A136 638 RULE-BASED STATISTICAL CALCULATIONS ON A DATABASE ABSTRACT(U) STANFORD UNIV CA DEPT OF COMPUTER SCIENCE N C ROWE JUN 83 STAN-CS-83-975 N00039-82-C-0250 2/2

UNCLASSIFIED F/G 5/2 NL

END


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

distributed numeric attribute of a relation. Let there be k such sets, of size m each, and let each first-order set cover a range of of R on the attribute. Suppose all queries that users ask refer to contiguous ranges of the attribute, ranges whose width is described by a probability density function p(w), w the width. We first consider the case where $w<R$. The odds that the user query width will overlap a boundary are w/R, and the standard deviation of the mean of a sample of w/R of m items is $\sqrt{(R/wm - 1/m)}$. Hence the average standard deviation for an estimate of the mean on this range is

$$\mu_{w<R}(w) = (1-w/R)\sqrt{(R/wm - 1/m)} + (w/R)\int_0^w \sqrt{(R/xm - 1/m)} + \sqrt{(R/(w-x)m - 1/m)}\, x\, dx$$

And for $w \geq R$:

$$\mu_{w\geq R}(w) = \mu_{w<R}(w \bmod R)/(w/R)$$

We can solve this numerically.

4.2.5. Choice of the value abstraction level

Another choice that must be made regarding the data of the database abstract is at what level of abstraction the statistics on the values should be tabulated (which may just be how much rounding for numeric values). Statistics can be estimated on the values of functions applied to sets (like region-of-nationality in queries 23 and 24 of section 2.1), but these are estimates, not exact values. Just as with sets, there is probably some optimum level of value aggregation, an "even" one in some sense, that we should use, and we can find it using the methods discussed in the last two sections. But note that the level of aggregation of values into first-order sets is something quite different, and that grain is usually too large to be appropriate for value abstraction. For instance, we may have only a dozen nationality-region first-order sets, but in answering queries on the frequency distribution of nationalities we need to treat every nationality as distinct.

4.2.6. Incomplete first-order abstracts

Of course, we need not include every statistic of every set that is included in the abstract. Some statistics may be quite accurately estimated from others, using the inter-statistic rules discussed in 3.4.2. An interesting problem which we have not investigated is to prune the search among the 2^s combinations of s statistics for the combination to include that minimizes some weighted sum of space and error. Heuristics are generally sufficient, as for instance the heuristics that the more useful statistic of the mean should always be preferred for storage to median, and the more useful mode frequency to the frequency of the second most common item[13].

4.2.7. Other sets in the database abstract

In principle, statistics on any arbitrary set can be put in the database abstract. The first-order sets should be designed to cover the domain of the database well, and thus should represent a good "general-purpose" collection of data. But if there is still room, other sets can be considered for inclusion. There are two main categories of such extensions of the abstract: systematic and cache.

4.2.7.1. Systematic extensions

A systematic extension includes statistics on every set of a given form. For instance, statistics on all the "second-order" intersection sets (intersections of two first-order sets), or only those second-order intersections that are bigger than a minimum size, or only those involving sets with atypical characteristics, like those with large outliers. Or we could have statistics on derived attributes, like statistics on all first-order sets with regard to logarithms of numeric attributes, or statistics on all first-order sets with regard to the products of corresponding values for two numeric attributes.

There are two difficulties to be faced with systematic extensions. First, estimates based on first-order information may be sufficiently close to the actual answer to make extra information not very useful. Second, the size and number of such extensions are often enormous. If there are m first-order sets, there are $O(m^2)$ intersections or unions of any two, $O(m^3)$ intersections or unions of any three, etc. If there are f numeric attributes, there are $O(f^2)$ virtual fields representing products of any two, etc. The number of ways of arranging and embedding intersections, unions, complements, and subsets also grows exponentially with their number. Thus systematic inclusion of other sets than first-order in the database abstract should be done with caution[14].

[13]We have done a few experiments with an incomplete abstract -- see section 6.8.

[14]We did some inconclusive experiments with a second-order intersection abstract on the first (merchant shipping database), discussed in section 6.9.

4.2.7.2. Caching extensions

The idea of caching extensions to the database abstract is to dynamically include statistics on sets particularly interesting to a user or users. If certain sets are being queried repeatedly, and in particular if a system knows that the user has been unsatisfied with an estimate and has gone to the full database to get an exact value for a statistic, important statistics on the set or sets in question can be placed in the abstract in anticipation of improving performance for future querying (statistics which may be just those retrieved anyway from a database). Cached statistics may be on quite small sets, and hence upwards inheritance (see section 3.4.5) (which need not arise at all with a systematic database abstract) is needed as well as downwards, necessitating special checks to avoid circularity. A few cached values may, in fact, be quite helpful to making conclusions on an abstract, because they can facilitate bounding from both directions on statistical estimates. For instance, suppose we know the longest American ship is 550 feet long, and the longest loaded American tanker currently in the Mediterranean is 420 feet long. Then the longest American tanker must be between 420 and 550 feet long. Thus it may be a good idea to include small sets (or even example items, sets of size 1) in a database abstract as well as the reasonably large first-order sets.

4.3. More on storage of the database abstract

We now discuss some issues not mentioned in the overview in section 1.7.1.

4.3.1. Format of the abstract

The database abstract should be stored by sets, with all statistics on the same set on the same physical page. (With about twenty statistics maximum per attribute, and 1000-word pages, this means at least 50 attributes can be tabulated for a set and still fit on a page, so there is no difficulty nearly all the time.) Since generally with our architecture, if one statistic is queried on a set, many others on that same set will be too, this is important.

Each set should have a header denoting which statistics on which fields are stored for it, and offsets to their locations. With this header information, data can be compressed (perhaps a factor of three for a typical database) and still be found, which is important is space is tight. A "base value" for each statistic on each attribute can be used, perhaps the value of the same statistic on the same attribute for the entire database relation, and the number stored can represent an offset to be added to this base value. Other base values can be used, like the average of the values for the two sets for an intersection of two sets. Ideas like these can significantly compress the size of the database abstract, particularly for normative statistics and well-behaved attributes.

It may be helpful to put special codes in the abstract having to do with inheritance, especially in regard to the priority of downwards vs. upwards vs. lateral inheritance estimates where all apply, since their appropriateness is quite data-dependent.

4.3.2. Indexing the database abstract

Clearly to avoid large searches one needs to index the sets of the database abstract. This can be a list of (set-name, page-pointer) pairs, sorted alphabetically on names. First-order sets are easy, but other sets have to be put in a canonical form before placement in the index. To use an example from our medical database, both (AND MALE (AND HITEMP HIPRED)) and (AND HITEMP (AND HIPRED MALE)) would be rearranged algebraically to the same pseudo-alphabetic form, (AND HIPRED (AND HITEMP MALE))[15]. One way to get a canonical form is to convert the query set expression to disjunctive normal form, alphabetize separately the terms in each conjuction, and then order the conjunctions by first term.

Another idea for storing the index to sets is the "tree" approach of keeping pointers to composite sets with their parents and relatives that are first-order sets. So next to the set MALE in the index have pointers to the sets "below" it, (NOT MALE), (AND MALE x), and (OR MALE x) where x is any other set. This helps when users query related sets frequently.

Efficiency of index storage and access speed should not be critical to system performance. Set names can be coded, and only two things need to be stored for each set in the index: set name code and pointer to its entry in the abstract. This can be compared to s*f things per set in the database abstract, s the number of statistics stored for the set, and f the number of attributes in the relation defining the set, which may be hundreds or thousands of items for interesting databases. So considering that the database abstract is intended to be, in most cases, considerably smaller than the database from which it is abstracted, the index to the database abstract will take up even less space, and can probably be stored in main memory. Even if it cannot, a good hashing scheme can mean that only one index page need ever be fetched in order to look up a set.

[15]MALE denotes visits by male patients, HITEMP denotes visits in which the patient had a high temperature, and HIPRED denotes visits in which the patient had high serum cholesterol.

4.4. Building the database abstract

Constructing the database abstract is a major enterprise.

4.4.1. Getting the abstract statistics from the database

The database abstract can be created on a single (though computationally expensive) pass through the database. Given a list of sets which it will tabulate statistics on, it examines each database page in turn, looking for members of those sets. Each time it finds an instance of a particular set, it updates registers for that statistic for that set; each set keeps its own distinct information. For set sizes this means incrementing a counter; for maxima and minima, comparison to and possible substitution in a register; and for means and standard deviations, adding to a sum register as well as incrementing a counter. Other statistics are trickier, generally requiring "reasonable-guess" assumptions as to the distribution of an attribute. For instance, median needs a likely range in which the median is likely to lie, points within which are kept in storage, and mode frequency needs both an estimated range of the frequency and an estimated range of the mode. Perhaps an earlier database abstract on the same database can be used to get the bounds.

4.4.2. Loading the database abstract

We surveyed this in section 1.8. If one does not wish to place guarantees on closeness of estimate (EST) values to actual values, loading of statistics computed on the database may be done in any order. But this closed-world guarantee can be quite powerful in narrowing the possible range on a query, and thereby improving estimation accuracy.

The guarantee will work if a partial ordering consistent with subquery invocation is imposed on all queries, and they are loaded in this order. Formally, we want a partial ordering (i.e., a noncyclic set of binary order relationships) for queries on our query language such that statistics on query set A are loaded before statistics on query set B if query A is a subquery needed in answering query B. This then guarantees that the accuracy of the estimate of B can't be changed by anything that is loaded after it, and if it satisfies 10% accuracy at loading, that accuracy will be maintained. So to give some examples, queries on A should be loaded before queries on (AND A B), queries on (AND A B) before queries on (OR A B), queries on a set with respect to the F attribute before queries on the same set with respect to (LOG F), queries on the maximum of a set in regard to an attribute before queries on the mean of the same set in regard to the same attribute, and so on.

Unfortunately, the simultaneous use of some interesting rule pairs does not satisfy a consistent partial ordering between queries (and not even in our prototype implementation, discussed in the next chapter). While it is useful to be able to bound a mean of a set by the maximum and minimum on that set, it is also sometimes useful to work in the other direction (as when maximum or minimum is unknown and mean is not, or when there happens to be an exact or very tight estimate of the mean and not the maximum or minimum), and bound the maximum, for instance, as the mean plus the product of the standard deviation and the square root of the set size -- that is, after section 3.5.1.2,

$$SUP(MAX(Q,F)) = MEAN(Q,F) + [SIGMA(Q,F) * \sqrt{SIZE(Q)}]$$

This "backwards" reasoning can arise often when arbitrary query answers are cached in the database abstract, and necessarily when upwards and lateral inheritance (see 3.4.5) are used. But we do not have to throw these kinds of rules away entirely in order to get the monotonicity of accuracy on loading. We can disenable them when making estimates of statistics for loading comparisons, but use them freely in actual answering of user queries. Thus to do loading we simply delete sufficient rules from our rule set to turn a cyclic graph into a tree, and we can pick and choose on intuitive criteria the rules we consider least important to delete sufficient to make a tree. See discussion in 5.5 for implementation details.

4.5. Updating the database abstract

If the database changes, we must update the database abstract to reflect it. As we have said, many statistical databases never updated and so this is not a central issue for our work. But there are systematic ways of proceeding if the database does change, without having to create the database abstract all over again.

4.5.1. The form of updates

Updates take two forms, inserts and deletes (changes to existing items may be considered a delete followed by an insert). A set of updates may thus be characterized by two lists of database tuples, an insert list and a delete list.

In order for the database abstract to use these lists, however, the items must be classified as to what first-order classes they belong to, to know which database abstract entries may have to be changed. Since first-order classes are defined as partitions on the values for particular attributes, this classification information may be thought of as a "generalized tuple". For instance, the inserted tuple

(PATIENT# = 33 SEX = F DISEASE.ACTIVITY = 2 TEMPERATURE = 37.5 CHOLESTEROL = 470 PREDNISONE = 20)

would correspond to the generalized tuple

(PATIENT.33 FEMALES MODERATELY.SICK NORMAL.TEMPERATURE HIGH.CHOLESTEROL MODERATE.PREDNISONE)

where each of those six names represents a first-order set in our rheumatology-database demonstration implementation (albeit the actual implementation names are more arcane). Since the information about how to classify tuples into first-order sets is not necessary for estimation on the database abstract, only in setup and update, this information should reside with the database, and the generalized tuple for each insert or delete needs to be transmitted along with the insert or delete, approximately doubling the amount of data that must be transmitted to the database abstract.

4.5.2. Modifying the database abstract vs. recomputing the statistic

Loading of the database abstract is computationally expensive, as we noted in 4.4.1, because every page and every record of the database must be examined. Recomputation of a single statistic on the same database is not much cheaper, considering that sets will be in most cases randomly scattered over many pages of a relation, and thus their retrieval will be paging-inefficient for the same reasons that random samples are paging-inefficient. Thus it is imperative to try to find "shortcuts", situations in which a known update (together with its generalized tuple) can be used to change the database abstract directly, without any statistical computation on the full database.

For example, suppose each attribute of a single-relation database is partitioned into 1000 ranges, each of which defines a first-order set (a somewhat larger number of partitions than most practical cases); and suppose there are 100 data records per page on the average for this database. Then one can use the same Poisson-distribution model for page accesses that was used for the 1-in-1000 random sampling discussed in section 1.3, and estimate the number of pages that contain a record of a random first-order set as one tenth of the database, not one thousandth. So to compute a single statistic on this database requires fetching about one tenth of the pages, not very good at all. Hence if there are only 10 different attributes in the database, each of which defines 1000 first-order sets, recomputing statistics on a database given even a *single* update will be nearly equivalent to fetching all the database pages, since an update will require recomputing statistics on every set mentioned in the generalized tuple. And things cannot be any better for more usual (smaller) numbers of range partitions.

(Of course, one can always get rid of the guarantee that the database abstract accurately reflects the database, and not bother to make updates at all. But absolute correspondence is essential to the validity of our reasoning about bounds.)

4.5.3. Formal derivation of update rules

There are a number of cases where the database abstract is easy to update when updates occur on the database. Rules for these cases can in many cases be derived logically from the other rules in our system by the following method. Denote the set of items to be inserted in a relation as the set INSERTS, and the set to be deleted as the set DELETES. Then the new database set can be described as

OR(AND(Q,NOT(DELETES)),INSERTS)

where Q is the old set. Oftentimes the inference rules applied to this will give an exact answer because we assume we have complete knowledge of the set of inserts and deletes available, and we may know a lot of statistics on Q. Also note we can assume the union is a disjoint union (there's no sense inserting into the database something already there), and disjoint unions lead to many exact rules. Similarly, an intersection with the complement of a set (i.e. a "set difference") is particularly nice since the set is known to be a subset of Q.

4.5.4. Rules for specific updates

Here are the rules for what to do for specific statistics when the database is updated.

- set size: just add SIZE(INSERTS) - SIZE(DELETES) to old database abstract value.

- mean: make new value

  [SIZE(Q)MEAN(Q) + SIZE(INSERTS)MEAN(INSERTS)
  - SIZE(DELETES)MEAN(DELETES)]
  / [SIZE(Q) + SIZE(INSERTS) - SIZE(DELETES)]

- maximum and minimum: for deletes, must go to database for a check if the deleted value is the old maximum or minimum, else make no change. For inserts, set value to larger (or smaller) of old value for the maximum (or minimum) of the inserts set.

- median: if the median of the deletes is equal to the median of the set, and the frequency of the median can be reasoned to be less than the number of deletes of it, then the median remains the same; otherwise must go to the full database to recompute the median. For inserts, one knows new median is bounded by the old median and the median of the inserts, hence if they are identical the new median is known exactly; otherwise must go to database to recompute. (Both solvable cases may be extended somewhat if item frequency information and minimum-possible gaps between distinct values are known.)

- mode: for inserts, compare two differences: between frequency of two most common items, and between the maximum net number of inserts for an item and the net number of inserts for the mode. Go to database to recompute when latter exceeds former. Do analogous thing for deletes.

- mode frequency: if the mode remains the same after the inserts and deletes according to the preceding rule, the new mode frequency is always the old mode frequency plus the number of inserts of the mode value and minus the number of deletes of that value. Otherwise, must go to the database to recompute except for a few special cases.

- number of distinct values in a set: must go to database to recompute unless one knows the exact values in the old set.

- MODEFREQ2, LEASTFREQ, MAXGAP, MINGAP, MAXEVENDEV, MINEVENDEV: must go to database to recompute.

Chapter 5
Implementation

All magic -- I repeat, all magic, with no exceptions whatsoever -- depends on the control of demons. By demons I mean specifically fallen angels. No lesser class can do a thing for you.

James Blish, Black Easter, *1968 (Avon, 1980)*

We discuss now some details of our prototype implementation, programmed in Interlisp [Teitelman 75] and run on the DEC-20 at ARPANET site SRI-AI. Section 5.2 discusses the implementation of the abstract, 5.3 the implementation of the rules. Control structure is covered in 5.4, the "backwards" rules in 5.5, and miscellaneous details in 5.6. Evaluation of this implementation is contained in chapter 6.

5.1. Databases used and statistics queried

We tested our prototype implementations on two databases. The first was a subset of the Stanford KBMS merchant shipping database containing data on four attributes (latitude, longitude, ship class, and registration nationality) of 605 tankers. The second was a subset of the database of the RX project at Stanford [Blum 82], containing six attributes (patient number, sex, disease-activity, temperature, measured serum cholesterol, and administered prednisone dosage) of 1000 clinic visits of 28 rheumatology patients. On the first database, 67 first-order sets were defined; on the second, 18. Two of the attributes of the first database were nonnumeric, and one of the second database. Since both databases could be expressed as single relations, we did not implement any join-manipulation rules.

5.2. The database abstract

In our experiments described in the next chapter an explicit database abstract was not created. We experimented with a variety of abstracts, both during development and evaluation, and containing both different sets and different statistics, and we did not want to have to create new abstracts for each experiment, or to edit an abstract thereby introducing the possibility of errors which, even when small, might ruin the experiment. Our usage of the abstract was considerably smaller than anticipated in a practical application, where there would be many more repeated queries, and thus the cost of building it is correspondingly more significant.

So instead of creating an explicit abstract, we access the database -- in both cases they were small enough to make this possible -- and dynamically compute statistics as needed. All such computed answers are cached in an answer list, so a subsequent repeat of the same query does not need recomputation -- which is important, because query answers are generally used in many different places. The answer list requires query sets and attributes to be in a canonical form. It also is currently unorganized and just searched linearly, but clearly if efficiency were important it could be indexed or hashed.

Most of our experiments discussed in the next chapter were done with "first-order" abstracts containing only statistics on the predefined first-order sets. However -- and this is one of the advantages of a virtual abstract over a real abstract -- it is easy to change the virtual abstract simply by modifying the procedure PRECOMPUTEDP that defines it. PRECOMPUTEDP takes three arguments -- a query set, an attribute, and a statistic -- and returns true if the answer to that particular query should be in the database abstract. We also have an analogous procedure GUARANTEEDP which returns true if a query has a 20% closed-world accuracy guarantee applied to it (see 1.8.2).

5.3. The rules

We do not store the rules as distinct entities, the way most production systems are documented, but in the "compiled" form of a decision tree [McDermott, Newell, and Moore 78, vanMelle 80, Leith 83] for efficiency. That is, the left sides (condition parts) of all rules are differentiated in COND clauses, calling upon subprocedures (possibly containing subprocedures as well, and so on) to handle all rules of a given type. For instance, our top-level procedure for handling boolean-expression sets first checks whether the outermost boolean operator in the set expression is an AND, OR, or NOT, and calls one of three subprocedures accordingly. Within each of these procedures further subprocedures are invoked, now depending on the kind of statistic being queried. Each of those may then invoke further subprocedures to handle particularly complex analysis situations. For instance, the "backwards" rules (see 5.5) for standard deviations are considerably more complicated than those for other statistics, and have been removed to a separate procedure.

There are both advantages and disadvantages to this particular rule implementation. It is fast and all control knowledge is "built in", making it clear what the program is doing at any given time. On the other hand, more complicated sorts of control (as resource-limited evaluation, time-sharing between different reasoning methods, etc.) are very difficult or impossible to implement. In addition, it harder to see in the compiled form just exactly what rules are being applied (or even how many there are), and it is harder to reason about patterns and analogies of rules.

Currently, about 400 rules are implemented: 65 statistic-statistic (section 3.4.2), 170 set decomposition (section 3.4.3), 150 attribute (field) decomposition (section 3.4.4), and 15 "backwards" rules (section 5.5), besides the four intraquadruple rules (section 3.4.1). Of the rule categories enumerated as the "computational dimension" in section 1-3, the following are not implemented: (1) relation decomposition rules (section 3.6.1), (2) rules for diagonal, rule, and general lateral inheritance (section 3.4.5), (3) rules for nulls (section 7.1.1), and (4) rules involving prototypes (section 3.6.2).

5.4. The control structure

Control of our program is centered in a single Lisp procedure, REFORM (figures 5-1, 5-2, and 5-3 -- a few inessential parts of the procedure have been omitted). This procedure organizes the answering of particular queries. It calls on subprocedures, which may eventually recursively call REFORM with subqueries, and so on.

We keep a cache (variable CACHE) of previously answered queries with their answers, which we check first before trying anything more (lines 7 through 11). (This cache is a Lisp association list and is searched sequentially, but it could be hashed for efficiency.) We also keep a stack (STK) of the subqueries currently invoked but not yet answered, to prevent infinite loops where a query is invoked as a subquery of itself (lines 12 through 19). If the same query is on the stack, NIL (meaning "I can't say anything about the answer") is returned; procedures QUICKMAX and QUICKMIN are explained in section 5.6.3, below.

```
(REFORM
 [LAMBDA (Q S F)
  (PROG (ANS NEWANS OLDCACHE)
     (COND
       ((NUMBERP F)
        (RETURN F)))
      [COND
       [(SASSOC (LIST Q S F)
            CACHE)
         (RETURN (CADR (SASSOC (LIST Q S F)
                    CACHE]
       ((MEMBER (LIST Q S F)
            STK)
        (RETURN (COND
              ((EQUAL S (QUOTE MAX))
               (QUICKMAX Q F))
             ((EQUAL S (QUOTE MIN))
               (QUICKMIN Q F))
             (T NIL]
        (SETQ STK (BAPPEND STK (LIST Q S F)))
```

Figure 5-1: Central control structure, part 1

Preliminaries disposed of, the program checks whether the value can be retrieved from the abstract, and if not, decides as per section 5.3 which top-level rule bundle procedures to invoke. This is handled in lines 25 through 46. These rules are specific to either the query syntax or the field syntax. The result is then filtered (lines 21 through 25) through quadruple-consistency rules (RESOLVE), statistic-statistic rules (FRESOLVE), and quadruple-consistency rules once again. If a percentage-accuracy applies to an inexact query estimate, it is then applied (lines 47 through 49). Then the cache is updated with the new answer (lines 50 through 59).

5.5. Backwards rules

We have some "backwards" rules in our implementation whose inclusion transforms the tree of subquery dependencies into a graph with cycles (see 5.5). These are handled in lines 47 through 58 of the REFORM procedure.

Backwards rules are necessary because one direction of inference between two statistics may work well for one query, and the opposite direction for another query. For instance, an upper bound on the number of distinct values in a set with respect to some attribute (SIZEUNIQUE) is the size of that set, corresponding to the situation where every item has a distinct value. This we designate a "forwards" inference rule in our system because set size is a statistic not predicated on an attribute, unlike the number of distinct values, and thus has only one value for a set instead of many, and thus is more valuable to have available for answering queries. The corresponding "backwards" rule is that a lower bound on a set size is the SIZEUNIQUE. Most of the time we will use the forward rule to estimate the number of distinct values, drawing upon methods like those demonstrated in section 1.5.1 to find a good upper bound on the size, which is then an upper bound on the SIZEUNIQUE. But we can also get additional bounds on set size by getting bounds on the SIZEUNIQUE with respect to a *different* attribute. In general we cannot predict which directions will be used in advance, and so we must allow for backwards rules if we are to get full power from our system[16].

There are problems with unrestricted use of backwards rules, however, even when they are eliminated from loading as discussed in section 4.4.2. Backwards rules create queries whose forwards subqueries include the same queries from which the backwards rules were initiated. Thus the original query will be invoked a second time, usually returning a NIL from line 19 in this

[16] In an informal survey, backwards rules applied about 10% to narrow the bounds on the size of the intersection of two sets in the medical database, so they are significant.

```
[SETQ ANS
 (SIMPLESTAT
  (RESOLVE
   (FRESOLVE
     [RESOLVE (COND
          ((PRECOMPUTEDP Q S F)
           (SIMULATE.ABSTRACT Q S F))
          ((NOT Q)
           0)
           ((NOT (ATOM F))
           (DOFIELD Q S F))
          ((ATOM Q)
           (DOATOM Q S F))
          ((EQUAL (CAR Q)
                (QUOTE OR))
           (DOOR Q S F))
          ((EQUAL (CAR Q)
               (QUOTE AND))
           (DOAND Q S F))
           ((EQUAL (CAR Q)
               (QUOTE NOT))
           (DONOT Q S F))
          (T [PRINT (APPEND (QUOTE (SORRY, I CANT HANDLE QUERIES
                WITH OPERATOR (LIST (CAR Q)) (QUOTE (IN THEM]
             (BOMBOUT]
    Q S F]
```

Figure 5-2: Central control structure, part 2

```
(COND
 ((GUARANTEEDP Q S F)
  (SETQ ANS (GUARANTEEANS ANS Q S F)))
  (SETQ CACHE (FAPPEND (LIST (LIST Q S F)
                    ANS)
                 CACHE)))
(SETQ OLDCACHE CACHE)
(SETQ NEWANS (RESOLVE (BRESOLVE ANS Q S F)))
  (SETQ CACHE (FAPPEND (LIST (LIST Q S F)
                    NEWANS)
                 (CDR OLDCACHE)))
(SETQ STK (BUTLAST STK))
(RETURN NEWANS])
```

Figure 5-3: Central control structure, part 3

invocation, thereby ignoring perhaps a lot of reasoning that had gone on about the query before the backwards rule was tried. Hence to take advantage of the work of previously invoked forwards rules we place in the cache a provisional estimate obtained by the work of the "forwards" rules before invoking the backwards rules (lines 47 through 49). If any backwards rules are found to change the estimate, the answer is so changed (lines 50 through 54), with any subqueries invoked in the course of applying the backwards rules deleted before finishing, since these answers are based on the superseded old estimate value for the top-level query.

5.6. Miscellaneous implementation issues

5.6.1. Optimization of UPPERBOUND and LOWERBOUND

Paging cost is going to be by far the predominant time cost in our system, since the operations performed in rules (with a few exceptions such as solving Diophantine equations) are quite simple and do not involve iteration. However, as rule systems become complicated (and especially with embedded quadruple algebra discussed in 3.3.4) some improvements may make a difference. At least, they can simplify debugging by eliminating useless information.

One major improvement in particular is straightforward to introduce, optimization of quadruple-element extraction. At many points in rules all we need is a bound, estimate, or error associated with the calculation of another statistic, and not the full quadruple itself. For instance, a lower bound on the number of distinct items in a set is the size of the set divided by the mode frequency; if the mode frequency is not known exactly, all that matters to the overall rule is that we know an upper bound (SUP) on it. We can think of this as a kind of optimization in the relational-calculus sense (cf. [Ullman 80], ch. 5), moving the quadruple-element extractors inside the query expressions as far as possible, e.g.

$$\mathrm{INF}(A/B) = \mathrm{INF}(A) / \mathrm{SUP}(B) \text{ if } A \geq 0,\ B > 0$$

The other rules given in 3.3.1 define the rest of the formal optimization.

In our implementation we have hand-optimized many of the rules this way, pushing extractors inside expressions. One disadvantage of this is that the optimized rules are harder to understand than the nonoptimized ones, since additional symbols are being inserted, and thus we trade off understandability for efficiency.

5.6.2. UPPERBOUND vs. probabilistic maximum

We also employ a related kind of simplification with bounds rules. Since we usually apply many different bounds rules to the same query, only a few will affect the result directly. In most cases we thus limit the effect of bounds to the SUP and INF parts of quadruples, and do not allow any effect on the EST and ERR parts, instead of using the technically correct probability-theory approach given in the footnote to section 3.3.2.

5.6.3. QUICKMAX and QUICKMIN

Normally we do no reasoning about a subquery if the same query is above it on the subquery stack. We make an important exception, however, for queries on maximum and minimum, since they are intimately related to the bounds given in many quadruples, and are queried repeatedly to answer most user questions. We thus can have a "cheap" way of evaluating them that incorporates only the most simple rules, which we call upon whenever the same query is above on the stack. Clearly such "cheap" evaluations could be done for other statistics too in other situations, but maximum and minimum seem to be by far the most important.

5.6.4. Binary decomposition of indefinite-number operands

Set intersections, set unions, attribute sums, and attribute products can involve arbitrary numbers of arguments. Our implementation for simplicity only handles binary combinations (e.g., AND(A,B,C) should be written as something like AND(A,AND(B,C))). The specific decomposition does not matter with most rules. For instance, MAX(AND(A,B,C)) is the larger of the three numbers MAX(A), MAX(B), and MAX(C), and these three numbers will be found regardless of whether the intersection is written AND(A,AND(B,C)), AND(AND(A,B),C), AND(B,AND(A,C)), or any other way. Similarly

$$EST(SIZE(AND(A,B,C))) = SIZE(A)*SIZE(B)*SIZE(C)/SIZE^2(UNIVERSE)$$

and this will be computed no matter how the binary decomposition is done. But the bounds on set intersection size obtained by reasoning about modes are inherently binary. For instance, SIZE(AND(A,AND(B,C))) may lead to a much better upper bound than SIZE(AND(AND(A,B),C)) if the mode frequency of the attribute defining C is much smaller than the mode frequency of the attribute defining A. To handle situations like this in our implementation, we generate every possible associative rearrangement of intersection or union (commutative rearrangements make no difference), and combine the results from each, following the methods of section 1.6.1: intersecting the bounds ranges, and taking the weighted average of the estimates (ESTs) and errors (ERRs).

5.6.5. A little domain knowledge

We have tried to isolate knowledge of the domain of the database in the abstract, but a few important things cannot be put there. These are things that are critical to system performance, but are basically intensional (true for any database state) instead of extensional (true for a particular database state) as the abstract. These include the names of first-order sets, what attributes they represent partitions on, the names of attributes, whether they are numeric or nonnumeric, and the definitions of functions on nonnumeric attributes. This data is kept in property lists.

5.6.6. User interface

Our implementation is only an experimental prototype, and we have not added much in the way of user aids. We do, however, paraphrase all query-language queries in simple embedded English clauses, and print this out before the query is executed, as an debugging aid for typographical and other obvious errors. When an answer is computed, it is formatted and printed with English explanations to make it clearer what it means. Values retrieved directly from the database abstract (PRECOMPUTEDP) and those to which a 20% accuracy guarantee applies (GUARANTEEDP) are so noted.

Chapter 6
Evaluation

"Is it a loss?" Rachael repeated. "I don't really know; I have no way to tell. How does it feel to have a child? How does it feel to be born, for that matter? We're not born; we don't grow up; instead of dying from illness or old age we wear out like ants. Ants again; that's what we are. Not you; I mean me. Chitinous reflex-machines who aren't really alive." She twisted her head to one side, said loudly, "I'm not alive! You're not going to bed with a woman. Don't be disappointed, okay? Have you ever made love to an android before?"

Philip K. Dick, Do Androids Dream of Electric Sheep, *1968 (Ballantine, 1982)*

We have claimed advantages of a new approach to estimation of statistics on a databases. We now back up these claims with quantitative experimental evidence of the comparative performance of our approach versus several simpler alternatives. We have validated performance on two quite different databases, demonstrating the generality of the ideas.

After an introduction in section 6.1, we explain our method for comparing answers and estimates in 6.2. Section 6.3 introduces the four basic control experiments needed to validate performance. Section 6.4 gives a set of results tables for the medical database. Section 6.5 discusses these results in regard to space and accuracy performance, and 6.6 in regard to time performance. The last three sections present results of three miscellaneous experiments: a "guaranteed-accuracy" abstract in 6.7, a partial first-order abstract in 6.8, and results for the merchant shipping database in 6.9.

6.1. Introduction

We claimed in section 1.3 that our "top-down" approach to statistical estimation has quite different advantages and disadvantages than the competing technique of random sampling, claims we would like to quantify. Our work is in the tradition of many rule-based "expert systems" developed in the field of artificial intelligence [Buchanan and Duda 82]. But unlike, say, a medical expert system designed to choose the most appropriate medical treatment for a patient, there is a quite rigorous way in which our statistical estimation expert can be evaluated: comparison of estimated numeric values with the actual values of the same statistic found by going back to the original data. We can thus quantify the disparity, and study the relative effectiveness of the estimation for different kinds of statistics, database sets, and attributes of those sets.

Evaluation of a statistical estimation system is tricky because there is a three-way tradeoff between space required, time required, and accuracy obtained. Plotted as a three-dimensional surface (see Figure 6.1), it is roughly a hyperboloid[17]. For a fixed level of accuracy, the curve is a hyperbola relating time vs. space. For less accuracy, the hyperbola moves closer to the vertical axis; moving this way "up" the surface is essentially what our system is trying to do. We would like to parameterize this surface to some degree.

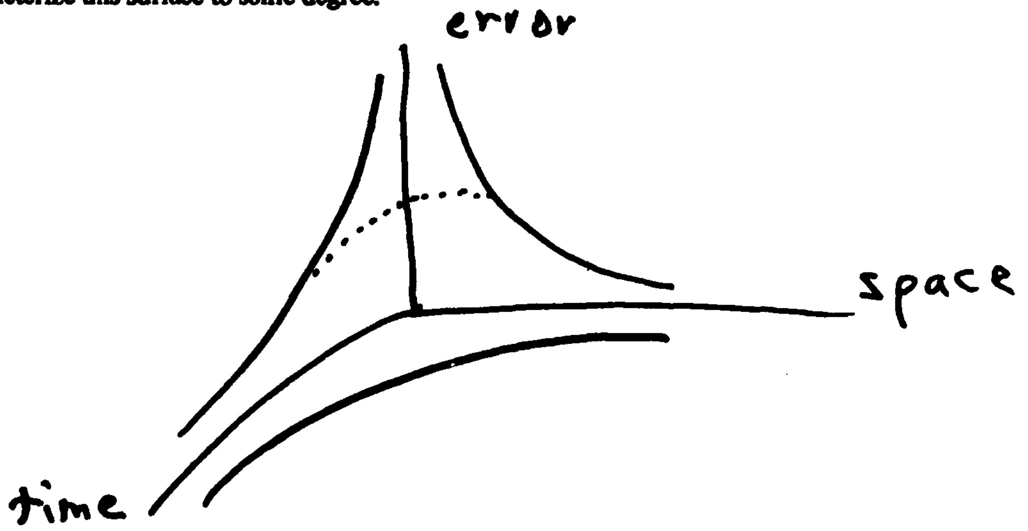


Figure 6-1: Space vs. time vs. accuracy

[17]Which we expect from the information-theoretic assumption that the total information quantity transmitted across a channel is constant, that is, SAT = K where S = space, A = accuracy, T = time, and K is some constant.

6.2. Comparing answers and estimates

To quantify "how close" an estimate is to an actual answer we use the number of consecutive high-order bits in common between the estimate and the answer, $-\log_2|(\text{est-ans})/\text{ans}|$. We wish to compare space and accuracy, and representing both in bits allows this. We could have used other metrics, as for instance an extremum of estimation rather than a normative summary of estimation, but we felt (1) extrema are harder to compare to space measurements, and (2) a good normative measure seems to correspond better to an intuitive performance assessment of an estimation system.

There are three problems with this bit-accuracy formula, however. First, the actual answer may be zero or negative; we ignored this, since much statistical data represents positive sums and counts anyway. (To handle this we would need to measure absolute accuracy or something else other than bits in common.) Second, the estimate may be exactly equal to the answer, giving an infinite number of bits in common; we handle this by putting a maximum on all such measures equal to the bit accuracy of the attribute whose statistics are being estimated. Third, the estimate may be greater than twice the answer, or less than half the answer, in which case the formula becomes negative; we handle this by arbitrarily rounding all negative results to zero.

We thus tabulate this performance metric on a series of random queries to our system. These abrupt changes for very close and very far apart values, however, are somewhat arbitrary, and so we also tabulate the breakdown of items in each of three categories (very close, reasonable, and very far) for a set of random queries.

Estimation performance depends on the database abstract as well as the inference rules, and so is database-dependent. In particular, it will not work well for data with complicated correlations between attributes. And since statistics on very small sets are not usually very meaningful, and subject to large variances, we impose the restriction that we only check our system performance on queries ten items or more in size. In addition, performance depends on the particular statistic queried, the form of the query set, and the query attribute. To avoid averaging these factors out over many queries we tabulate performance separately for major categories.

6.3. Control experiments

In order to demonstrate that our estimation approach is advantageous we must have a standard of comparison, a "control" experiment. Matters are complex because there are at least four such controls. We present them in approximate order of increasing challenge to our methods.

1. answering the query from a database abstract without using any inference rules (the "null rules" control)

2. answering the query with a minimal abstract (just statistics on each relation as a whole), but a full set of inference rules (the "null abstract" control)

3. running the query on the full original database, calculating the exact answer (the "full database" control)

4. "upwards" inference from a random sample the same size as a particular database abstract (the "sampling" control)

Our system must perform at least as well as all four of these in order for it to be judged a "success". By "at least as well" we mean that if any two of the three factors of space, time, and accuracy are held constant, performance will be better in regard to the third factor. To put it in terms of figure 6.1, the hyperboloid representing behavior for the experimental scheme must be "below" the hyperboloid representing behavior for the control scheme for a reasonable range of parameters. In some cases, only one factor needs to be held constant. For instance, for the first and fourth controls above, we shall show that both time and accuracy are better when space is held constant.

Restrictions that the database, abstract, or sample be of a certain size or type, are fair when stated explicitly. Significance can be checked by the standard deviation of the bit accuracy and usual hypothesis-testing methods.

6.4. Some results

We developed an implementation of our ideas originally for four attributes of a small subset of the merchant shipping database of the KBMS project at Stanford, doing debugging and preliminary evaluation with this data. To demonstrate the generality of our ideas more convincingly we needed a different database, and we chose a random subset of the database of the RX project at Stanford [Blum 82], itself a subset of the ARAMIS (American Rheumatology Association Medical Information System) database of information about rheumatology patients. As we discussed in section 4.2.2, we chose six attributes to analyze for 28 patients with a total of 1000 visit records: patient number, sex, disease activity level, temperature, measured cholesterol, and administered prednisone. (Occasional missing values for the last four attributes were filled with values on previous visits.) We created a vocabulary of 18 named (or "first-order" sets) whose statistics would be stored in the abstract, representing partitions into two parts on sex, four parts on disease activity, and three parts for the other four attributes.

We summarize our results in tables. Each table contains several subtables representing different experiments. Each experiment is numbered, and contains three lines of result data. For explanation of the format, see figure 6-2.

Table 6-3 shows results for statistics on the temperature attribute, table 6-4 the prednisone dosage, table 6-5 results for two arithmetic operations between attributes, and table 6-6 results for set unions. Experiments tested particular query set forms, attributes or "fields", and abstracts. The same query sets were tested for each of the six statistics. "Exact" means that exact rules give a certain value for the answer, not an estimate; this is implies in our table a 10.0 bits of accuracy, a 10-0-0 answer breakdown, and .00 range narrowing. For set intersections, only those larger than 10 items were used for tests, since statistics on smaller sets fluctuate widely, and the statistics are not particularly significant anyway. For set unions, because of time constraints, we disenabled the "backwards" reasoning or relaxation-style analysis used in all the other tests here; hence results are not as good.

Each experiment involved testing of estimation performance for ten random queries on six statistics: count (set size), mean, max (maximum), sigma (standard deviation), median, and modefreq (mode frequency). Results for each statistic are tabulated separately, and are spaced horizontally across the page in six columns, in that order.

Results for each statistic in an experiment are summarized in seven numbers presented on three lines, in this format:

<bits of accuracy>(<standard deviation of accuracy>)
<# of exact answers>-<# of reasonable estimates>-<# of poor estimates>
<average range narrowing>(<standard deviation of narrowing>)

where:

- <bits of accuracy> is the average number of bits in common between the estimate (EST) and the actual statistic value, in ten random queries, computed according to the formula in section 6.2. 10 bits is assumed the maximum accuracy for all these experiments, since it is the accuracy of the numeric data.
- <standard deviation of accuracy> is the standard deviation of those numbers for the ten queries
- <# of exact answers> is the number of queries, in the ten, that can be answered to at least 10 bits of accuracy, the accuracy of the data
- <# of reasonable estimates> is the number of estimates, in the ten, that were not near-exact, but no worse than twice the actual answer or half the actual answer
- <# of poor estimates> is the number of estimates, in the ten, that were more than twice the actual answer of half the actual answer
- <average range narrowing> is the average ratio, in ten random queries, of the range between the bounds on the estimate to the possible range of that statistic
- <standard deviation of the narrowing> is the standard deviation of the preceding in ten random queries

Figure 6-2: Format of results for each experiment

count	mean	max	sigma	modefreq	median
Experiment 1: statistics on the intersection of two first-order sets, with respect to the temperature attribute, for a first-order abstract					
8.24(3.6)	9.09(1.2)	7.23(2.0)	3.41(1.9)	3.22(2.9)	8.17(1.3)
8-1-1	1-9-0	0-10-0	0-10-0	1-9-0	0-10-0
.01(.01)	.32(.19)	.28(.16)	.10(.07)	.07(.10)	.29(.16)
Experiment 2: same as experiment #1 but for null abstract					
.34(.70)	7.00(1.6)	3.95(1.1)	1.87(2.0)	.02(.04)	6.59(1.0)
0-2-8	0-10-0	0-10-0	0-7-3	0-1-9	0-10-0
1.0	1.0	1.0	1.0	1.0	1.0
Experiment 3: statistics on a first-order set, for the square root of temperature, first-order abstract					
exact	10.0(0)	exact	2.59(2.0)	exact	exact
	10-0-0		0-8-2		
	.002(.01)		.12(.09)		
Experiment 4: same as #3 but for null abstract					
.50(.90)	8.97(.9)	5.82(1.4)	0(0)	.52(1.4)	8.17(1.2)
0-3-7	1-9-0	0-10-0	0-0-10	0-2-8	0-10-0
1.0	1.0	1.0	1.0	1.0	1.0
Experiment 5: statistics on a first-order set, for the square of temperature, first-order abstract					
exact	exact	exact	7.38(1.7)	exact	exact
			0-10-0		
			.24(.14)		
Experiment 6: same as #5 but for null abstract					
1.21(1.7)	7.26(1.3)	3.93(1.3)	2.33(1.5)	.09(.18)	6.33(1.1)
0-4-6	1-9-0	0-10-0	0-8-2	0-3-7	0-10-0
1.0	1.0	1.0	1.0	1.0	1.0

Figure 6-3: Esiimation of statistics on patient temperatures

count	mean	max	sigma	modefreq	median
Experiment 7: statistics on the intersection of two first-order sets, with respect to the prednisone dosage attribute, for a first-order abstract					
6.27(3.9)	3.51(1.9)	3.56(3.1)	1.49(1.6)	2.57(3.2)	4.99(3.4)
5-5-0	0-10-0	0-8-2	0-7-3	1-7-2	3-7-0
.03(.04)	.34(.31)	.34(.31)	.14(.08)	.08(.05)	.34(.31)
Experiment 8: the same as #7 but for a null abstract					
.73(1.0)	2.20(2.1)	1.90(3.3)	1.40(1.6)	.40(.92)	6.25(4.6)
0-4-6	0-7-3	0-6-4	0-7-3	0-2-8	0-8-2
1.0	1.0	1.0	1.0	1.0	1.0
Experiment 9: statistics on the square root of prednisone, first-order abstract					
exact	7.77(1.4)	exact	2.51(3.0)	exact	exact
	0-10-0		0-6-4		
	.05(.04)		.13(.05)		
Experiment 10: same as #9 but for a null abstract					
.76(.73)	4.11(2.6)	2.72(3.7)	0(0)	.79(1.6)	6.59(4.2)
0-6-4	0-10-0	0-6-4	0-0-10	0-2-8	6-4-0
1.0	1.0	1.0	1.0	1.0	1.0
Experiment 11: statistics on the square of prednisone, first-order abstract					
exact	exact	exact	1.90(1.1)	exact	exact
			0-10-0		
			.07(.06)		
Experiment 12: same as #11 but for a null abstract					
.86(1.4)	1.22(1.5)	2.99(4.6)	.16(.5)	.54(1.2)	6.14(4.7)
0-3-7	0-7-3	0-3-7	0-1-9	0-2-8	6-3-1
1.0	1.0	1.0	1.0	1.0	1.0

Figure 6-4: Estimation of statistics on prednisone dosages

count	mean	max	sigma	modefreq	median
Experiment 13: statistics on a first-order set, with respect to the sum of corresponding values for prednisone and cholesterol, with first-order abstract					
exact	exact	5.59(1.8)	5.52(2.2)	.65(.8)	5.97(3.0)
		1-9-0	1-9-0	0-6-4	3-7-0
		.11(.08)	.04(.03)	.61(.89)	.08(.07)
Experiment 14: same as #13 but with null abstract					
.38(1.1)	4.45(1.7)	1.50(2.2)	.62(1.2)	.39(1.2)	6.46(3.1)
0-1-9	0-10-0	0-7-3	0-3-7	0-1-9	4-6-0
1.0	1.0	1.0	1.0	1.0	1.0
Experiment 15: statistics on a first-order set, with respect to the product of corresponding values for prednisone and cholesterol, with first-order abstract					
exact	6.76(2.1)	3.42(3.4)	1.25(1.8)	.15(.3)	3.63(3.3)
	2-8-0	2-8-0	0-5-5	0-3-7	2-8-0
	.57(.36)	.34(.31)	.27(.18)	.30(.49)	.07(.05)
Experiment 16: same as #15 but for null abstract					
1.11(1.1)	1.99(1.6)	.20(.40)	.29(.77)	.40(.76)	0(0)
0-6-4	0-8-2	0-2-8	0-2-8	0-2-8	0-0-10
1.0	1.0	1.0	1.0	1.0	1.0

Figure 6-5: Some virtual-attribute statistics

count	mean	max	sigma	modefreq	median
Experiment 17: statistics on the union of two first-order sets, for the temperature attribute, with first-order abstract					
6.84(2.4)	6.56(2.4)	exact	3.54(4.3)	7.12(2.5)	6.26(2.1)
3-7-0	3-7-0		3-3-4	4-6-0	1-9-0
.11(.14)	.37(.36)		.28(.25)	.07(.11)	.42(.35)
Experiment 18: same as #17 but for null abstract					
.39(.9)	7.29(1.4)	4.45(1.1)	.87(1.6)	.23(.50)	7.00(1.3)
0-3-7	0-10-0	0-10-0	0-4-6	0-2-8	0-10-0
1.0	1.0	1.0	1.0	1.0	1.0
Experiment 19: statistics on the union of two first-order sets, for the prednisone attribute, with first-order abstract					
7.91(1.9)	6.59(2.6)	exact	6.63(2.5)	6.49(2.8)	2.19(3.3)
3-7-0	3-7-0		3-7-0	3-7-0	1-6-3
.05(.09)	.01(.02)		.04(.04)	.03(.04)	.22(.25)
Experiment 20: same as #19 but with null abstract					
1.12(1.4)	3.58(1.9)	5.25(4.8)	3.02(3.2)	.92(.95)	0(0)
0-8-2	0-10-0	0-10-0	0-9-1	0-6-4	0-0-10
1.0	1.0	1.0	1.0	1.0	1.0

Figure 6-6: Results for set unions, without relaxation

6.5. Discussion: space and accuracy

Section 6.3 gave four separate control experiments we must compare performance against. Our experiments do not provide a complete comparison to each, in part because of time and space limitations[18], but they do cover most of the issues. The basic philosophy of this evaluation is determination of "the value of rules" in the style of [Michie 76].

6.5.1. Control 1: Abstract, no rules

Clearly we can answer many more queries with rules on an abstract than without. An abstract can only contain contain a finite number of query answers, whereas rules can give statistics on arbitrarily large intersections and unions of sets in the abstract. The space for the rules is negligible compared to the size of the abstract because (a) rules can be coded efficiently since they contain few different symbols, and (b) we are interested primarily in large data sets where the abstract (as well as the database itself) is likely to be considerably larger than the rule storage.

6.5.2. Control 2: Rules, no abstract

We study this by experiment. For "no abstract" we still mean to include statistics on entire relations, information which it seems reasonable to assume is accessible to a user without the computer -- technically, a "null abstract". These conditions apply to the even-numbered rows in our first four tables. As one can see by comparing the figures with those for corresponding queries with a first-order abstract (experiment 2 with experiment 1, 4 with 3, 6 with 5, etc.), performance is usually significantly better. We can quantify the level of significance by the standard deviations given in parentheses on the first lines of the entries.

6.5.3. Control 3: calculation on full database

The third control experiment is getting the exact answer by running the query on the full database. The data is 1000 tuples with 6 attributes, a total of $1000 * 6 * 16 = 96,000$ bits. The first-order abstract used in the experiments consists of 19 first-order sets plus the universe, with 14 statistics tabulated for numeric attributes and 5 for nonnumeric, for a total of $5*14 + 5 = 75$

[18] We used about 50 hours of CPU time on a DEC-20 at SRI International to perform this evaluation, coming close to the space limitations of single-user Interlisp in the process. 90% of the time expended was calculation of the database abstract values when needed from the actual data.

attributes, each with 10 bits of accuracy, for a total of 19 * 75 * 10 = 14,250 bits, or about 14.8% of the size of the database. (We ignore here the size of the program to manipulate the database abstract, as it is fixed in size independent of the database and database abstract, and its rules can be coded highly efficiently in few bits if desired.)

The main difference, however, is between the exact answers given by full-database querying and the limited accuracy of estimates. The tests we have run give average bits of accuracy for particular query forms. If multiply this by the number of possible queries of a given type, and sum up over all query types, we can get an estimate of a "virtual database size" due to the inclusion of inference rules along with the abstract. Of course there are an infinite number of queries since intersections and unions can be embedded arbitrarily deep, but one can set reasonable limits on query size (or better yet, weight query types as per their frequency of occurrence). As an example, consider just our estimates of the intersections of two sets. There are about 18 * 15 = 270 such sets, and the six statistics computed on these sets in the first row of table 6-3 cover 8.24 + 9.09 + 7.23 + 3.41 + 3.22 + 8.17 = 39.36 bits on the average per set, so there is virtual storage for about 270 * 39.36 = 10,600 bits, representing near to a doubling of the database abstract size. Similar figures can be summed for all common query forms. The total sum represents how well the rules are extending a given database abstract, and may be roughly compared to the size of the original database.

6.5.4. Control 4: Random sampling

The fourth and last control experiment is to extrapolate from a random sample the same size as the database abstract. We studied this experimentally by constructing a random sample the same size as our first-order (18-set) database abstract, 148 sample items out of 1000 in this case, and inferring upwards from statistics on the sample to statistics of the population. Results are contained in tables 6-7 and 6-8. Experiments should be compared as follows:

- experiment 21 with experiment 1
- experiment 22 with experiment 7
- experiment 23 with experiment 17
- experiment 24 with experiment 19
- experiment 25 with experiment 3
- experiment 26 with experiment 9

- experiment 27 with experiment 5
- experiment 28 with experiment 11
- experiment 29 with experiment 13
- experiment 30 with experiment 15

Our rule-based method is about the same or better in most comparisons, while at the same time being likely to have much better access time in terms of page retrievals for all but very small databases, as discussed in section 1.3, and while avoiding the "brittleness" mentioned there in regard to sets of related queries.

count	mean	max	sigma	modefreq	median
Experiment 21: statistics on the intersection of two sets and the temperature attribute					
2.00(1.8)	8.14(2.9)	7.38(3.3)	3.07(2.8)	2.51(3.8)	8.55(3.2)
0-8-2	1-8-1	5-4-1	0-7-3	0-7-3	8-1-1
Experiment 22: statistics on the intersection of two sets and the prednisone attribute					
2.30(1.4)	4.15(2.6)	3.35(4.4)	1.80(1.4)	2.36(2.1)	4.74(4.3)
0-9-1	0-9-1	3-3-4	0-7-3	0-7-3	4-5-1
Experiment 23: statistics on the union of two sets and the temperature attribute					
2.60(1.3)	9.51(.6)	4.73(1.1)	2.94(1.5)	2.62(2.3)	8.96(.7)
0-10-0	0-10-0	0-10-0	0-10-0	0-9-1	3-7-0
Experiment 24: statistics on the union of two sets and the prednisone attribute					
3.29(1.8)	3.84(2.2)	2.50(3.8)	3.52(1.5)	2.74(1.2)	8.45(3.1)
0-10-0	0-10-0	2-6-2	0-10-0	0-10-0	8-2-0
Experiment 25: statistics on a first-order set, of the square root of the temperature attribute					
2.57(1.5)	9.09(1.2)	6.78(2.2)	1.45(1.4)	3.08(1.9)	8.80(1.3)
0-9-1	0-10-0	3-7-0	0-7-3	0-10-0	3-7-0
Experiment 26: statistics on a first-order set, of the square root of the prednisone attribute					
2.75(1.7)	4.59(2.2)	3.29(3.4)	2.65(1.9)	2.38(1.5)	8.27(3.5)
0-9-1	0-10-0	2-7-1	0-8-2	0-10-0	8-2-0

Figure 6-7: Results for extrapolation from a random sample, page 1

count	mean	max	sigma	modefreq	median
Experiment 27: statistics on a first-order set, of the square of the temperature attribute					
2.82(1.6)	7.95(1.5)	6.48(3.0)	3.44(2.4)	2.59(2.9)	8.07(1.8)
0-10-0	0-10-0	4-6-0	0-9-1	0-8-2	4-6-0
Experiment 28: statistics on a first-order set, of the square of the prednisone attribute					
2.86(1.1)	3.28(1.8)	6.0(4.9)	2.00(1.5)	2.71(.6)	8.28(3.4)
0-10-0	0-10-0	6-0-4	0-10-0	0-10-0	8-2-0
Experiment 29: statistics on a first-order set, of the sum of corresponding values for prednisone and cholesterol					
1.75(1.6)	4.66(2.8)	4.34(3.9)	1.33(1.8)	1.64(1.8)	5.99(3.8)
0-6-4	0-8-2	3-5-2	0-4-6	0-7-3	4-4-2
Experiment 30: statistics on a first-order set, of the product of corresponding values for prednisone and cholesterol					
2.34(1.6)	3.50(2.0)	2.69(3.8)	2.22(1.6)	1.98(1.5)	5.07(3.6)
0-9-1	0-9-1	2-4-4	0-8-2	0-7-3	3-6-1

Figure 6-8: Results for extrapolation from a random sample, page 2

6.6. Discussion: time

These limited experiments do not well address the tradeoff between time and the other factors of space and accuracy, because significant advantages do not accrue unless the database is several orders of magnitude larger. A primary motivation for the database abstract architecture is the improvements in paging performance over random sampling and full-database-access methods, and 1000 sextuples should be easy to fit into most any computer's primary memory. But if we pretend that we have very limited primary memory and that all large datasets (database abstract and random sample as well as full-database tuples) are kept in pages (say 1000 16-bit words) in secondary storage, we can make the following comparisons for each of the four control experiments. (We assume, as with most large databases, that page accesses are the only significant time cost.)

6.6.1. Control 1: abstract, no rules

We assume rules can be coded efficiently and kept in core, hence they add no paging cost. Hence there is no difference in time.

6.6.2. Control 2: rules, no abstract

This alternative does obviate paging of the abstract, but that is only one page (14,250 bits = 890 16-bit words) for these experiments. For larger abstracts we assume all statistics on the same set are placed on the same page, and so an upper bound on the number of page accesses is the number of different sets queried. This number is constant for all queries of a given form. For intersections of two sets it is three: the first set, the second set, and their intersection. For unions it is four: all the preceding plus the union of the two sets. For unary and binary operations on simple attributes it is one since only the set actually queried need be accessed. So the number of page accesses needed to estimate a statistic with the database abstract is a small constant independent of the size of the abstract, rule set, or database.

6.6.3. Control 3: calculation of answer on full database

The database is stored on 1000 * 6 / 1000 = 6 pages. Even if there is an index pointing to every tuple of a given set, unless the sets are very small (say, 10 items or less) it is likely that at least one item of the set is on every page, except for the rare case where the database is clustered with respect to the partitioning that defines the set. Hence all six pages will need to be fetched nearly all the time, whereas only one page in these experiments, or a small constant number of pages in general, will need to be fetched to use the database abstract, a clear cost savings during query answering.

The database abstract does require setting up, however, which in turn requires accessing these very same six pages. But we only wish to use the database abstract architecture when setup work is small compared to query answering, and setup cost can be amortized to insignificance over a large number of queries. Setup can be made page-efficient, too, by implementation as a single-pass algorithm through the database.

6.6.4. Control 4: random sampling

For setup, this has essentially the same paging costs as the use of the abstract, since for reasonably-sized samples and more than just a few tuples per page, nearly every page must be retrieved for at least one tuple. In this particular case, the random sample is 148 items, and the odds are very high that each of the six pages will be represented. For a larger database with p items per page, and a random sample of size m of the database, the number of pages looked at will be $p(1-e^{-m/p})$ from a Poisson model as mentioned in section 1.3. Since our approach must always look at every page during setup, the net paging advantage of random sampling during setup is $(1-e^{-m/p})$. For $m=p$, i.e. the number of sample points being equal to the number of database pages, this is only a savings of $e^{-1}=36.8\%$ over our approach, and for most databases this represents a very small random sample, too small to be useful.

Once the random sample and the database abstract are created, they both fit into the same amount of space, and the same number of pages. But answering queries with the sample will likely require accessing most pages of it, because even if there is an index (which may require additional paging to obtain), one has similar paging inefficiencies with random placement of records as with random sampling of the full database. Thus as the size of the sample increases, the necessary paging to answer queries will increase nearly proportionately. At the same time, answering queries with the abstract will require a fixed number of page accesses (bounded by the total number of pages) depending on the form of the query, as discussed in section 6.6.2. In addition, new random samples usually need to be fetched from the database if a user is interested in another data set, whereas the database abstract is general-purpose. In a distributed architecture where the database abstract and/or random sample are separated from the database by a low-bandwidth connection, these additional fetches may be intolerable.

6.7. A guaranteed-accuracy system

Table 6-9 gives results for a first-order order abstract and a 20% "guarantee" on second-order intersections (i.e., statistics on the intersections of any two first-order sets), following the discussion of section 1.8.2 and the example of item 9 of section 1.5.1. That is, any statistic on an intersection of two sets not explicitly in the abstract (virtual abstract, actually) is assumed to be estimatable to within 20% by rules (leaving out the "backwards" rules discussed in section 5.5, for the reasons discussed there). All values not estimatable to that precision are explicitly put into the abstract on loading. Thus extra effort in the loading of the abstract trades off with additional query accuracy, which may or may not be worth it.

For these tests we estimate a statistic on a second-order intersection needed to be stored in the database abstract 10% of the time for counts, 40% of the time for means, 20% for maxima, 70% for sigmas (standard deviations), 70% for mode frequencies, and 60% for medians. Since there are about 270 second-order sets, we estimate the size of this "1 and 1/2" order abstract as:

size of first order entries + size of second order entries
= 14,250 bits +
[270 sets * 5 attributes * 14 statistics * 10 bits of accuracy maximum
* (.1 + .4 + .2 + .7 + .7 + .6)/6]
= 14,250 + (189,000 * .45) = 98,950 bits

which is about the size of the database, and hence not worth the small amount of extra accuracy obtained (as one can see from comparing experiment 31 with experiment 1 in figure 6-3, and experiment 32 with experiment 7 in figure 6-4).

count	mean	max	sigma	modefreq	median
Experiment 31: Statistics on the intersection of two sets, with respect to the temperature attribute, for a "one and half"-order abstract containing statistics on all first-order sets, plus statistics on other sets as needed to provided a 20% accuracy guarantee on all second-order set statistics.					
8.76(2.5)	8.60(1.1)	8.71(1.9)	5.18(2.5)	7.30(3.5)	8.02(1.0)
8-2-0	0-10-0	0-10-0	2-8-0	6-4-0	0-10-0
.01(.02)	.28(.18)	.27(.19)	.01(.01)	.27(.20)	
Experiment 32: same as #31, except for the prednisone attribute					
8.68(2.2)	4.95(2.2)	7.14(2.2)	6.40(3.0)	8.38(2.8)	7.06(3.2)
7-3-0	1-9-0	3-7-0	4-6-0	7-3-0	5-5-0
.01(.01)	.06(.04)	.08(.08)	.03(.02)	.01(.01)	.02(.02)

Figure 6-9: Results for second-order ±20% guarantees

6.8. Results for a partial first-order abstract

We also ran a few tests on a "partial" first-order database abstract consisting of only the statistics SIZE, MAX, MIN, MAXEVENDEV, MINEVENDEV, and SIZEUNIQUE on the first-order sets and the universe. As may be seen in figure 6-10, results are not good; compare experiment 33 with experiment 1, 34 with 3, and 35 with 13. Thus it seems important to have thorough statistics on first-order sets, at least for this database.

count	mean	max	sigma	modefreq	median
Experiment 33: statistics on the intersection of two first-order sets with respect to the temperature attribute, for an abstract containing only SIZE, MAX, MIN, MAXEVENDEV, MINEVENDEV, and SIZEUNIQUE statistics for first-order sets					
.40(.7)	6.72(1.3)	3.46(1.0)	1.01(1.1)	0(0)	6.25(.8)
0-3-7	0-10-0	0-10-0	0-6-4	0-0-10	0-10-0
1.0(0)	1.0	1.0	1.0	1.0	1.0
Experiment 34: statistics on first-order sets with respect to the square root of the temperature attribute, for an abstract containing only SIZE, MAX, MIN, MAXEVENDEV, MINEVENDEV, and SIZEUNIQUE statistics for first-order sets					
.77(.9)	9.49(.8)	5.04(.9)	0(0)	.64(1.4)	9.16(.7)
0-5-5	1-9-0	0-10-0	0-0-10	0-3-7	1-9-0
1.0(0)	.88(0)	.88(0)	.88(0)	1.0(0)	.88(0)
Experiment 35: statistics on first-order sets with respect to the sum of the corresponding prednisone and cholesterol values, for an abstract containing only SIZE, MAX, MIN, MAXEVENDEV, MINEVENDEV, and SIZEUNIQUE statistics for first-order sets					
exact	exact	6.70(2.0)	6.50(2.3)	.27(.5)	7.79(2.4)
		2-8-0	2-8-0	0-4-6	5-5-0
		.04(.05)	.02(.02)	.44(.62)	.02(.03)

Figure 6-10: Estimation with a partial first-order abstract

6.9. The merchant shipping database

We also ran some evaluation tests on the merchant shipping database used for system development. The field was chosen randomly; between latitude and longitude for numeric statistics (MEAN, MAX, SIGMA, and MEDIAN), and between those two, ship nationality, and ship type for nonnumeric (SIZE and MODEFREQ). This test was done previous to all the other tests discussed in this chapter, and not all the rules used in the others were employed, and a few infrequently-appearing bugs remained to be found. Results are in figures 6-11 and 6-12. Note the format is different from the other results in this chapter: the top number is the bits of accuracy, and the next line the breakdown of answer types in ten random queries.

count	mean	max	sigma	modefreq	median
Experiment 36: statistics on the intersection of two first-order sets, for a random attribute, and a first-order abstract					
53.2	35.7	41.7	21.9	32.8	35.1
2-0-8	0-10-0	0-10-0	0-9-1	1-8-1	0-10-0
Experiment 37: same as #35 but with null abstract					
0	35.0	23.6	10.1	1.6	28.9
0-0-10	0-10-0	0-10-0	0-7-3	0-2-8	1-9-0
Experiment 38: same as #35 but with union instead of intersection					
96.2	70.0	exact	76.2	66.3	36.5
8-2-0	4-5-1		6-4-0	5-5-0	1-9-0
Experiment 39: same as #37 but with null abstract					
0	0.8	50.0	1.2	10.0	30.8
0-0-10	0-4-6	0-8-2	0-3-7	2-0-8	2-3-5
Experiment 40: same as #35 but on the intersection of 3 first-order sets					
14.9	15.9	18.5	8.6	8.6	24.2
0-8-2	0-9-1	0-8-2	0-5-5	0-6-4	0-9-1
Experiment 41: same as #39 but with second-order intersections in abstract too					
46.3	23.7	25.6	27.3	10.7	30.4
4-6-0	0-9-1	0-8-2	0-9-1	0-6-4	0-10-0
Experiment 42: same as #39 but with second-order 20% guarantee abstract					
15.8	29.5	15.9	3.3	15.9	27.7
0-7-3	0-10-0	0-8-2	0-2-8	0-6-4	0-10-0

Figure 6-11: Some results for the merchant shipping database

count	mean	max	sigma	modefreq	median
Experiment 43: statistics on a first-order set with respect to the sum of latitude and longitude, for a first-order abstract					
exact	exact	47.8	18.4	1.8	58.6
		1-9-0	0-9-1	0-4-6	2-8-0
Experiment 44: same as #42 but for a null abstract					
exact	32.2	21.2	14.2	1.9	47.4
	0-10-0	0-10-0	0-8-2	0-2-8	0-10-0
Experiment 45: same as #42 but for the product of latitude and longitude					
exact	31.8	15.3	23.9	1.9	3.9
	0-10-0	0-9-1	0-10-0	0-6-4	0-1-9
Experiment 46: same as #44 but for a null abstract					
exact	20.3	10.5	12.0	1.1	0
	0-10-0	0-6-4	0-7-3	0-1-9	0-0-10
Experiment 47: statistics on a first-order set with respect to a random unary operator, for a first-order abstract					
exact	84.0	exact	32.9	exact	exact
	4-6-0		0-9-1		
Experiment 48: same as #46 but for a null abstract					
exact	92.1	exact	36.9	exact	exact
	7-3-0		2-8-0		

Figure 6-12: More results for the merchant shipping database

Chapter 7
Extensions and further applications

Money, that's what it took. A little money -- a tenth of the cash they lavished on the Computer Science Department, say -- and he could have parapsychology really on the move. Going places. They were doing it elsewhere: Professor Fether in Chicago was testing precognition in hippos; the Russians claimed a breakthrough on the ouija board to Lenin; the ghost labs in California were fast building a solid reputation. But here, a standstill, a frozen landscape. Nobody in the entire field had ever heard of the University of Minnetonka.

John Sladek, Roderick, *Pocket Books, 1980*

This thesis opens up a broad area for future research. We now mention some directions that could be followed. Section 7.1 mentions some possible extensions to the work, and 7.2 new applications of the completed work.

7.1. Extensions

7.1.1. Further statistics and rules

We have examined fourteen statistics on numeric attributes and five on nonnumeric in this work. Clearly we could look at more, such as other order statistics, correlations (but see section 3.1.3), tuple abstractions, possible values, results of smoothing, and various Gestalt properties of a set not shared by its members.

There are many rules that we could study but have not as yet, such as rules for correlations, causations, rules for intensional knowledge, rules for prototypes, dependencies, quantifiers, and nulls. Nulls are particularly easy to handle by extending the database abstract with additional sets. In defining a first-order set with respect to some attribute, nulls for that attribute can be interpreted as items with any value possibility. Hence any set can be viewed as the disjoint union of items known definitely to be in it with a set of items having nulls for the defining attribute. If the size of the former set is known, we then have a lower bound on the total set size as this number, and an upper bound as this number plus the number of nulls. Other statistics can be handled similarly by this disjoint-union model.

As we mentioned in section 1-4, many of the methods we used by hand to derive rules could be automated with symbolic-algebra systems doing much of the work. Rules for new cases and statistics could be derived whenever a need for a particular category is foreseen.

7.1.2. Augmented algebra of uncertainty

We could include information on possible values for statistics, extending quadruples into quintuples. This requires modification of all the interval-algebra and normal-variable-algebra operations, and extra specifications in every rule. But this sort of reasoning may be very useful, as it is quite different from any other reasoning in our system, and likely to be "orthogonal" to (independent of) other results.

Another idea is to allow disjoint ranges in quadruples; for instance, to express a range as the union of two disjoint ranges. Such a situation arises upon dividing by an interval containing zero, which we disallow in our implementation (see 3.3.1), but other rules may also lead to disjoint ranges, as when the standard deviation for a set is almost equal to half the range, implying that values are clustered around the maxima and minima.

Explicit mention of exceptions with the quadruples may also be fruitful to explore. Much statistical data contains a few extreme "outliers", points very atypical, that are often eliminated before analysis proceeds to avoid biases in calculating means and standard deviations. Keeping these exceptions explicitly with quadruples allows operating on them separately in calculations.

7.1.3. More sophisticated control structure

Our current control structure is meant to be as simple and explicit as possible, to facilitate emphasis of the rules themselves. Many things could be done to add speed and/or flexibility. Partial orderings defining defaulting for estimate rules could be derived automatically from analysis of the rules, by proving that certain rules logically derive from certain other rules. Branch-and-bound methods could be used to decide in advance if a category of bounds rules could help on an answer. Relaxation methods, both numeric and nonnumeric, could be used in a more systematic and breadth-first way to handle what is now forced to be an awkward and inefficient depth-first tree traversal, further complicated by the ad-hoc way in which we handle backwards rules.

The database abstract can in principle contain any statistics at all on the database. But we have not exploited this much in our database abstracts since this requires careful planning of how to use downwards, upwards, and lateral inheritance (see section 3.4.5) simultaneously, and also creates problems for loading (see 4.4.2), because it creates cycles in the ordering of queries.

7.1.4. Hardware implementations

Special-purpose hardware could improve performance of our system. Obtaining the database abstract is computationally expensive, and special devices for computing the basic aggregate statistics on the database in the one-pass simultaneous-tracking method described in 4.4.1 might be very helpful, perhaps as components in disk drives. Such devices would also improve answer speed for arbitrary statistical queries on the database, but our database abstract approach would still have a place for those users who do not have the money or space for the special hardware and full database, or the time to wait for an exact answer.

As for our system itself, paging is also the major cost, and thus a read-only-memory database abstract might significantly improve performance; this could be quite cost-effective for much-used databases like the U.S. Census. Rule computation time is less significant, but with more complex rules (e.g. Diophantine analysis) and bigger rule sets (e.g. if automatic derivation of rules is done) it might start to matter. Rules could be coded in a read-only-memory themselves, independent of the database, in a highly compact code which could then be decoded for retrieval. Note that much of the answering of queries involves reasoning along many independent lines (bounds reasoning in particular), and this could be implemented with multiprocessing if speed were important. Processors could write subquery results onto a "blackboard" as they found them, and these results could be used or modified by other processors working on other lines of reasoning. Relaxation methods would need to be supported, for one processor might get different results depending on when another processor working on a relevant subproblem posts its answer.

7.1.5. Better evaluation

Due to time (CPU) and space (in Interlisp) constraints, we have only been able to run tests on two small databases. (For data analysis, however, their size is quite typical.) We need to run detailed studies on a database at least ten times larger, perhaps one hundred times or more, before we can confirm the large savings in page faults we have predicted.

We also need more random queries per category, since the dispersion of results in many categories is large. Perhaps we could try to cluster results into categories of relative accuracy, and analyze the queries for which particular amounts of accuracy are obtained. We also need to run "matched pairs" of queries on experiments and controls, since the ten random queries generated for each may be different.

The criterion for evaluating the effectiveness of the bounds, the "average narrowing" provided, is rather arbitrary and unsatisfactory. What we really want is the decrease in the entropy of a statistic, in the style of [Haq 75]. But it is difficult to say just what our state of initial knowledge is; whether, for instance, it includes all the possible statistics on the universe, or only some, or also statistics on common sets, and so on, and just which "obvious" modes of inference are included. The matter deserves further study.

7.2. Further applications of this work

7.2.1. Cooperative user interfaces

An original motivation for this work was the providing of more "cooperative" answers to user queries when an exact answer would take much time, namely a quick estimate answer. A number of presentation methods could improve the cooperativeness of this. "Bargaining" could be done where costs of the actual answer and estimates (perhaps several) could be shown and compared. Answer quadruples could be expressed graphically in various ways [Tukey 77] so that users do not have to read and interpret numbers. Or quadruples could be expressed in "fuzzy" English, e.g. "the answer is pretty big". If it can be inferred by one way or another (including being told) that the user is interested in a statistic in order to answer a yes/no question -- e.g., "Are there tankers over 700 feet?" -- the question can be answered directly. In addition, the system could be much more informative about its operation, noting and paraphrasing rules that were significant in deriving an answer.

7.2.2. Query optimization

Our methods can be used to improve performance of a database system.

7.2.2.1. Access paths

Estimates of statistics and bounds on them are very useful in "optimizing" performance of a database system ([Ullman 80], ch. 6). Set size estimates in particular are equivalent to "selectivities" used in deciding between alternative access paths, as in choosing an algorithm to perform a join. Other statistics can indirectly help to estimate set sizes. For instance, the mean and standard deviation of an attribute of a set suggest how many values might lie in some range of that attribute (i.e., the size of a first-order set defined on the values). Bounds on set sizes can be useful too because they allow *guarantees* that the optimal access path is being followed in lieu of guesses, for the worst case, no matter how good the guesses.

An interesting idea to explore is making the first-order database abstract sets correspond to actual physical pages in the database. Then the statistics on these pages could be used in lieu of (or in addition to) an index to help decide whether the page was relevant to the answer of a particular query.

7.2.2.2. Implicit optimization in rules

The statistical inference rules we have developed also represent a kind of optimization themselves, namely optimization of statistical queries. What is being optimized is accuracy of the answer, or perhaps ability to answer from the abstract (assuming that more complex sets are rarer in the abstract) not processing time. For instance, the rule

MAX(OR(A,B)) = LARGEROF(MAX(A),MAX(B))

looks very much like the optimization methods for relational algebra discussed in [Ullman 80] applied to the new operator, MAX. Unfortunately, there are only a few of these exact rules, and it is arguable whether one can still use the word "optimization" for inexact rules, which give query forms which are not equivalent, but the principle is the same.

7.2.2.3. Optimization of statistical processing

Inferred bounds can also help optimize the calculation of actual order statistics and frequency-distribution statistics statistics on the database. (Thus our methods can be of use even in a system used only for high-accuracy hypothesis testing, or "confirmatory data analysis".) For instance, if one knows absolute bounds on the median, one can absolutely bound the amount of storage space need for a one-pass calculation of it. Similarly for the mode; and one can get additional help if bounds are known on the mode frequency or other frequencies, allowing pruning of items from the possibilities list. In general, bounds are useful because they can support non-backtracking control structures for calculation of order and frequency statistics, by putting good absolute bounds on the amount of space required for the calculation.

7.2.3. Reasoning about uncertainty

The quadruple algebra provides an apparently new way of reasoning about uncertainty. It could be related to other methods. For instance, Dempsey-Shafer theory [Barnett 81] reasons about bounds on probabilities of discrete events, and this might be combined with our own reasoning about events characterized by continuous-valued variables. Fuzzy set theory [Zadeh 79], however, is fundamentally incompatible with our methods, and we provide a comprehensive alternative to it. We believe our work raises serious questions for the continued use of fuzzy set theory because our approach, with much the same functionality although different emphases, is much more rigorously justified in terms of simple reasoning and maximum-entropy theory with inequalities.

One aspect of our quadruple representation of uncertainty and the associated inference rules is that it provides, for apparently the first time, a complete knowledge representation scheme based entirely on the concept of a set (for previous attempts, see [Brachman and Smith 80]). There is no epistemological need to distinguish concepts representing individuals from concepts representing sets; what have been called individuals are just sets with SIZE equal to 1, and special "small-set" inference rules can be used to reason about other small sets (e.g. if you know the maximum, minimum, and mean of a set of these items or less, you know what those items must be). But it is essential to distinguish, as in other work in artificial intelligence, the *intension* of a set from its *extension*, two things that arise from quite different sorts of knowledge, though their constraints can be manipulated through quadruple algebra the same way; our work mainly applies to the latter.

7.2.4. Statistical database inference security

One accidental byproduct of our work is a testbed for inference security research with databases [Denning 82], in that we have a system that concludes as much as it can about data from a limited number of statistical aggregate values. Such a system has never existed before; previous work has examined at only a few rules, or only single rules, and not the possibly synergistic interactions of very different kinds of rules. For inference security work the bounds rules are most of interest, but the estimate rules can also be used for "reasonable-guess" inference compromise. For research of this sort, we would probably want to include rules to handle rare but strong-bounds special cases that are not cost-effective for statistical estimation, i.e. we should optimize the rule set for power (i.e. accuracy) and not speed and space.

The database abstract itself can be considered as a protection mechanism for a database, if queries are only allowed to it and not to the full database. The idea is similar to that of "partitioning" [Yu and Chin 77]: predefining a partitioning of the records of a database and forcing all queries to be on these predefined partitions markedly simplifies the analysis of inference security.

7.2.5. Knowledge-based compression

Our database abstract and rules can be seen as a sophisticated kind of data compression of a database. While the compression factor is intended to be large and the abstract not information-preserving, this is not intrinsic, and a database abstract could be designed that did capture all of a highly redundant database, plus just our exact rules (though other rules can occasionally lead to exact conclusions too through a progressive narrowing of possible values).

Chapter 8
Conclusions

In the thesis we have presented a new approach to low-cost estimation of statistics on a large computer database. We provided an overview in chapter 1. This method reasons "top-down" from a precomputed set of statistics on a particular database (a "database abstract") using a set of inference rules. We enumerated important potential advantages over random sampling for a large class of applications.

Chapter 2 then demonstrated performance of our implementation. We showed our quadruple estimates for a variety of queries on our two test databases, a merchant shipping one and a medical one. Answer accuracy varied a good deal with the statistic, the query set, and the attribute of that set, but many of our answers were quite reasonable.

Chapters 3 and 4 then went into the details of how the system works. Chapter 3 covered the rules, elaborating on the rule taxonomy given in chapter 1. We explained what statistics we answer questions about, and defined the "quadruple algebra" needed to generalize arithmetic for our system. We elaborated the different situations in which rules apply, the "computational" dimension, and then the different methods by which rules can be derived and justified. Chapter 4 then went on to explain the database abstract, covering its format, loading, and updating.

Chapter 5 and 6 addressed our particular implementation of these ideas. Chapter 5 covered a few specific decisions we made in order to get a working demonstration system. Chapter 6 then analyzed performance for a set of randomly chosen queries to our system, broken down into categories by statistic queried, query form, attribute form, and kind of database abstract. Using appropriate metrics we found that our system had as good or better accuracy for queries as extrapolation from a random sample, while at the same time having the potential for large savings in paging time. The database chosen for these evaluation experiments was a different one from the database on which the system was developed, thus demonstrating a degree of portability.

Chapter 7 then mentioned some ways our system could be improved, and some applications of it to some different problems.

Statistics is a new domain for rule-based expert systems, and much remains to be explored. The success of our interdisciplinary integration of research in databases, statistics, and artificial intelligence suggests that productive work in this area will continue to be interdisciplinary, and will probably involve information theory and operations research as well. We believe that the availability of rigorous evaluation methods for our domain is very important, channeling the direction of research in the best directions and decreasing unsupported speculation.

There are interesting parallels between our domain and that of symbolic algebra [Tobey 71]. Both use a diverse class of of methods to attack continuous-function mathematics. Originally emphasizing heuristic search methods from artificial intelligence, symbolic algebra has become increasingly mathematical as new analysis methods have been found for important subproblems. It may be that the same thing will happen to our domain, as the mathematics is better understood.

The idea of providing for the first time a comprehensive alternative to random sampling, for estimating characteristics of a large data population at low cost, is very appealing. Random sampling is so important for such a variety of important problems that a real alternative with different advantages and disadvantages can be quite significant. Perhaps we can anticipate a future in which both random samples and database-abstract are available for many large databases, and one can choose between them according to needs. The possibilities are exciting.

Appendix A
Definitions of the statistics used

First:

- Let $x_1, x_2, \dots x_n$ be a set of data values, denoted "x_is".
- Let y_is be the x_is sorted in increasing order.
- Let f_is be the number of occurrences (frequency) of the corresponding elements of y_i.

Let "$\max_i$" denote the maximum value of something over all i, and "Σ_i" the sum of something over all i. Then:

- the SIZE of the set is n
- the MAX of the set is y_n
- the MIN of the set is y_1
- the MEAN of the set is $\Sigma_i x_i / n$
- the SIGMA (standard deviation) of the set is $\Sigma_i x_i^2 - [\Sigma_i x_i]^2$
- the MEDIAN of the set is $x_{\lfloor (n+1)/2 \rfloor}$
- the SIZEUNIQUE (number of distinct items) is the number of distinct values in x_i
- the MODEFREQ of the set is $\max_i(f_i)$
- the MODE of the set is $y_{\min_i(f_i = \mathrm{MODEFREQ}_i)}$
- the MODFFREQ2 of the set is $\max_i[f_i$, where $f_i \neq \mathrm{MODEFREQ}_i]$
- the LEASTFREQ of the set is $\min_i(f_i)$
- the MAXGAP of the set is $\max_i[y_{i+1} - y_i$, where $i \neq n$ and $y_{i+1} \neq y_i]$

- the MINGAP of the set is $\min_i[y_{i+1}-y_i$, where $i \neq n$ and $y_{i+1} \neq y_i]$
- the MAXEVENDEV of the set is $\max_i[(x_i-x_1)/(x_n-x_1) - (i-1)/(n-1)]$
- the MINEVENDEV of the set is $\min_i[(x_i-x_1)/(x_n-x_1) - (i-1)/(n-1)]$
- the KEYS of the set are all attributes or groups of attributes taken together that uniquely distinguish a tuple within this set (not including keys that logically follow from other keys by appending arbitrary additional attributes to a known key)

Appendix B
Mapping the query language to SQL

Our set-oriented query language is very close in functionality to standard database query languages. As an example, we show how to map it to the language SQL, alias SEQUEL [Chamberlin et al 76].

A table is needed to decode the meaning of the names denoting first-order sets (see section 4.2.2), consisting of four entries: the name (SETNAME), what relation it refers to (RELATION), what field (attribute) of that relation (FIELD), and a logical specification of what range of values corresponds to the name (VALUES). The latter can have two forms: a list of possible values, or (for numeric values) an upper and lower bound.

The query:

"What is the STAT of the set SETNAME with respect to attribute ATTRIB?"

translates into SQL as:

```
select CODE<STAT>(ATTRIB)
from RELATION<SETNAME>
where FIELD<SETNAME> in VALUES<SETNAME>
```

for VALUES that is a list, or

```
select CODE<STAT>(ATTRIB)
from RELATION<SETNAME>
where VALUES-LOWBOUND<SETNAME> <= FIELD<SETNAME>
    and FIELD<SETNAME> <= VALUES-HIBOUND<SETNAME>
```

for VALUES a range (e.g. "length > 500" becomes "FIELD<SETNAME> > 500"). Angular brackets <> denote instantiating arguments that will be removed before execution. RELATION<SETNAME> indicates the lookup of the RELATION for SETNAME, VALUES<SETNAME> and FIELD<SETNAME> analogously; these represent text strings inserted into the SQL code before it is interpreted. CODE<STAT> denotes the appropriate SQL name for the statistic STAT; many statistics used in this work are not in SQL, but conceptually they're simple to add.

Some cleverness can be used to code a list of values permissible for a set. Abstraction symbols can be used to stand for sets of basic values, e.g. "tankers" can be defined to have values ALI, SEA, MIS, etc., and each of those can be defined by lists of actual database values ~~for the~~ ship type attribute.

As for Boolean combinations of sets, these have to be composed on the above "where" clauses, i.e.

```
where [where-clause for SETNAME1] AND [where-clause for SETNAME2]
where [where-clause for SETNAME1] OR [where-clause for SETNAME2]
where NOT [where-clause for SETNAME]
```

Joins can be represented as a nested clause as an additional AND term in the "where" clause; and need a special table coding what fields are common to any two relations, and these fields are used in the "where <fields> in" header for the nested clause. Alternatively, we can treat the join as a relation itself. For instance,

"Give the mean of length of the American ships."

where length is a property of ship type, in a separate relation from nationality, could be represented as:

```
select mean(length)
from join of ships and shiptype on type
where nationality in {us}
```

Appendix C
Some rule examples

We give here a few examples of rules, following our two taxonomies for the computational (figure 1-3) and derivational (figure 1-4) dimensions, respectively.

C.1. Examples for the computational dimension

1. intraquadruple rule: an estimate of the mean of a set for an attribute is the average of the maximum and the minimum for that attribute.

2. statistic-statistic rule: an upper bound on the median of a set is the maximum of the set.

3. row (set) decomposition rules:

    a. subset inheritance: an upper bound on the standard deviation of a subset is the standard deviation of the set times the square root of the ratio of the set size to the subset size.

    b. set intersection: an estimate of the mean of the intersection of two sets is the weighted average of their means, where the weighting is the reciprocal of their set sizes.

    c. set union: a lower bound on the median of the union of two disjoint sets is the smaller of the medians of the two sets.

    d. set complement: a lower bound on the mode frequency of the complement of a set is the difference of the mode frequency of the item universe (the set of all items in the database) and the mode frequency of the set.

4. column (attribute) decomposition rules:

    a. unary 1-to-1 operations: an upper bound on the mean of the logarithm of an attribute is the logarithm of the mean.

    b. other unary operations: the exact value for the maximum of the absolute value of an attribute is the larger of the absolute value of the maximum and the absolute value of the minimum.

c. binary operations: the exact value for the mean of the sum of corresponding values for two attributes is the sum of the means on the two attributes.

d. vectorization: a lower bound on the number of distinct values of the vector of corresponding values for two attributes is the larger of the number of distinct values for each attribute.

e. operations with constants: the exact value for the mode of K times an attribute is K times the mode.

5. relation (join) decomposition rule: an upper bound on the size of the join of two relations on an attribute is the size of one attribute times the mode frequency of the other on the attribute.

6. unusual inheritance rules:

   a. upwards inheritance: a lower bound on the maximum of a set is the maximum of a subset.

   b. lateral inheritance: an estimate of the mode of a set is the mode of some other subset of a superset of the first set.

   c. diagonal inheritance: an estimate of the mean of a subset is the weighted difference of the mean of a superset and the mean of a different subset of that superset, where the weighting is the size of the superset and subset respectively.

   d. attribute-hierarchy inheritance: a lower bound on the number of distinct combinations of certain attributes for a set is the number of distinct combinations of only some of those attributes.

   e. rule inheritance: any special rule that applies to finding an upper bound on the maximum of a set also applies to finding an upper bound on the maximum of a subset.

7. query rearrangement rule: any statistic on the union of a set with the intersection of that same set with another set is just the statistic on the set.

8. the closed-world rule: if the policy followed was that a statistic on set S and attribute A was only loaded into the abstract if it was estimable within K%, then the absence of that statistic from the abstract means that the estimate now obtainable from rules is within K%.

C.2. Examples for the derivational dimension

1. basic mathematics rule: an upper bound on the size of the intersection of two sets is the smaller of the sets.

2. probability and statistics rules:

   a. definitions: the mode frequency of a set is an upper bound on the frequency of any other item.

   b. theorems: the standard error associated with the estimate of the mean of a subset as the mean of the set is the standard deviation of the set times the square root of the difference of the reciprocals of the subset size and the set size.

   c. extrema of definitions and theorems:

      i. bounds on values: an upper bound on the standard deviation is the larger of the distances of the mean from the maximum and the mean from the minimum.

      ii. independence assumption: an estimate of the size of the intersection of two sets is the product of the two set sizes divided by the size of the universe (all items of that given type).

      iii. linearity assumption: a lower bound on the mean of the logarithm of values is α times the logarithm of the minimum of the values plus $(1-\alpha)$ times the logarithm of the maximum of the values, where α is the ratio of the distance of the mean of the values from the minimum to the distance of the maximum from the minimum.

      iv. nonlinear optimization: an upper bound on the standard deviation of a set is the geometric mean of the distance of the mean from the maximum, and the distance of the mean from the minimum.

      v. entropy maximization: knowing the mean, maximum, and minimum of a set, but in the absence of further information, the standard deviation can be estimated by fitting to a truncated exponential distribution and computing that distribution's standard deviation. (See footnote to section 3.4.1 for the mathematical details.)

3. database theory rules:

   a. functional dependencies: the mode frequency for values of an attribute which is an extensional key for a set is 1.

   b. inference compromise:

      i. small samples and trackers: if set A contains all of set B plus one other item, then the value of that item for some numeric attribute is the mean of A minus the ratio of the difference of the two means over the size of B.

ii. exploiting uniqueness: if set A and set B have the same median for some attribute, and that attribute is an extensional key, then the intersection of set A and set B has at least one member, the item with that median value for that attribute.

iii. Diophantine equations: if the values for some set attribute are all divisible by some number K, then the product of the set size and the mean for that attribute must also be divisible by K.

4. new rules from old:

a. rule composition: an upper bound on the proportion of members of the intersection of set A with set B, within some universe, is the smaller of the proportions of A and B within that universe.

b. rule rearrangement: an upper bound on the size of a set is the product, for any attribute, of the mode frequency and the number of distinct values.

c. theorem proving: an upper bound on the mode frequency of a set for an attribute is one more than the difference of the size of the set and its number of distinct values for the attribute.

d. analogies: a lower bound on the minimum of a set with respect to the difference of corresponding values for two attributes is the difference between the minimum for the first attribute and the maximum for the second.

5. prototype rule: representing all sets as points in some dimensional space, the median of a set is the weighted average of the medians of all other sets, where the weighting is the reciprocal of the Euclidean distance of the point representing the set from the points representing each of the others.

Bibliography

[Arnold 74] Barry C. Arnold.
Schwarz, Regression, and Extreme Deviance.
The American Statistician 28(1):22-23, February, 1974.

[Barnett 81] Jeffrey A. Barnett.
Computational Methods for a Maathematical Theory of Evidence.
In *Proceedings*, pages 868-875. Seventh International Joint Conference on Artificial Intelligence, Augusut, 1981.

[Bates et al 82] Doug Bates, Haran Boral, and David J. DeWitt, .
A Framework for Research in Database Management for Statistical Analysis.
In *Proceedings*, pages 69-78. ACM SIGMOD International Conference on Management of Data, June, 1982.

[Blum 82] Robert L. Blum.
Discovery, Confirmation, and Incorporation of Causal Relationships from a Large Time-Oriented Clincial Data Base: The RX Project.
Computers and Biomedical Research 15:164-187, 1982.

[Brachman and Smith 80]
R. J. Brachman and Brian C. Smith, eds.
Special Issue on Knowledge Representation.
SIGART Newsletter (70), February, 1980.

[Brachman and Israel 81]
R. J. Brachman and D. J. Israel.
KL-ONE Overview and Philosophy.
In W. A. Woods (editor), *Research in Knowledge Represenation for Natural Language Understanding: Report No. 4785*, pages 5-26. Bolt Beranek and Newman, 1981.

[Buchanan and Duda 82]
Bruce G. Buchanan and Richard O. Duda.
Principles of Rule-Based Expert Systems.
In M. Yovits (editor), *Advances in Computers*, . Academic Press, New York, 1982.

[Carbonell 80] Jaime G. Carbonell.
Default Reasoning and Inheritance Mechanisms on Type Hierarchies.
In *Proceedings*, pages 107-109. Workshop on Data Abstraction, Databases, and Conceptual Modelling, Pingree Park CO, June, 1980.

[Chamberlin et al 76]
D. D. Chamberlin et al.
SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control.
IBM Journal of Research 20(6):560-575, 1976.

[Cheeseman 83] Peter Cheeseman.
A Method of Computing Generalized Bayesian Probability Values for Expert Systems.
In *Proceedings of the Eighth Meeting.* International Joint Conference on Artificial Intelligence, 1983.

[Christodoulakis 81]
Stavros Christodoulakis.
Estimating Selectivities in Data Bases.
Technical Report CSRG-136, University of Toronto, December, 1981.

[Cole and Morrison 82]
A. J. Cole and R. Morrison.
Triplex: A System for Interval Arithmetic.
Software -- Practice and Experience 12:341-350, 198[illegible].

[Cox 80] Lawrence H. Cox.
Suppression Methodology and Statistical Disclosure Control.
Journal of the American Statistical Association 75(370):377-385, June, 1980.

[Davis and King 76]
R. Davis and J. King.
An Overview of Production Systems.
In E. W. Elcock and D. Michie (editors), *Machine Intelligence 8*, pages 300-334. Wiley, New York, 1976.

[Denning 82] D. E. Denning.
Cryptography and Data Security.
Addison-Wesley, Reading, MA, 1982.

[Denning 83] Dorothy E. Denning.
A Security Model for the Statistical Database Problem.
In *Proceedings.* Second LBL Workshop on Statistical Database Management, September, 1983.

[Denning et al 79]
D. E. Denning, P. J. Denning, and M. D. Schwartz.
The Tracker: a Threat to Statistical Database Security.
ACM Transactions on Database Systems 4(1):76-96, March, 1979.

[Fahmy and Proschan 81]
Salwa Fahmy and Frank Proschan.
Bounds on Differeoces of Order Statistics.
The American Statistician 35(1):46-47, February, 1981.

[Fernandez ct al 81]
Eduardo B. Fernandez, Rita C. Summers, and Christopher Wood.
Database Security and Integrity.
Addison-Wesley, Reading MA, 1981.

[Gill, Murray, and Wright 81]
Philip E. Gill, Walter Murray, and Margaret H. Wright.
Practical Optimization.
Academic Press, New York, 1981.

[Givone 70] Donald D. Givone.
Introduction to Switching Circuit Theory.
McGraw-Hill, New York, 1970.

[Graham, Yao, and Yao 80]
R. L. Graham, A. C. Yao, and F. F. Yao.
Information Bounds are Weak in the Shortest Distance Problem.
Journal of the Association for Computing Machinery 27(3):425-444, July, 1980.

[Haq 75] M. I. Haq.
Insuring Individual's Privacy from Statistical Data Base Users.
In *Proceedings*, pages 941-946. National Computer Conference, 1975.

[Hart 82] Peter E. Hart.
Directions for AI in the Eighties.
ACM SIGART Newsletter (79):11-16, January, 1982.

[Haugen 68] Edward B. Haugen.
Probabilistic Approaches to Design.
Wiley, New York, 1968.

[Kahneman and Tversky 82]
Daniel Kahneman and Amos Tversky.
On the Study of Statistical Intuitions.
Cognition 11:123-141, 1982.

[King 81] Jonathan J. King.
Query Optimization by Semantic Reasoning.
Technical Report STAN-CS-81-857, Stanford University Computer Science Department, May, 1981.

[Kocnig and Paige 81]
S. Koenig and R. Paige.
A Transformational Framework for the Automatic Control of Derived Data.
In *Proceedings of the 7th Meeting*, pages 306-318. International Conference on Very Large Data Bases, Cannes, France, 1981.

[Korn and Korn 68]
Granino A. Korn and Theresa M. Korn.
Mathematical Handbook for Scientists and Engineers.
McGraw-Hill, New York, 1968, pages 332-345chapter 11.

[Ku and Kullback 68]
H. H. Ku and S. Kullback.
Interaction in Multidimensional Contingency Tables: An Information Theoretic Approach.
Journal of Research of the National Bureau of Standards -- Mathematical Sciences 72B(3):159-199, July-September, 1968.

[Ku and Kullback 74]
Harry H. Ku and Solomon Kullback.
Loglinear Models in Contingency Table Analysis.
The American Statistician 28(4):115-122, November, 1974.

[Lawler 64] Eugene L. Lawler.
An Approach to Multilevel Boolean Minimization.
Journal of the Association for Computing Machinery 11(3):283-295, July, 1964.

[Leith 83] Philip Leith.
Hierarchically Structured Production Rules.
The Computer Journal 26(1):1-5, 1983.

[Lenat 82] Douglas B. Lenat.
The Nature of Heuristics.
Artificial Intelligence 19:189-249, 1982.

[Lenat et al 79] D. B. Lenat, F. Hayes-Roth, and P. Klahr.
Cognitive Economy.
Working Paper HPP-79-15, Stanford University Heuristic Programming Project, June, 1979.

[Lloyd's 82] Lloyds Register.
Lloyd's Register of Shipping: Statistical Tables 1982.
London, U. K., 1982.

[McDermott, Newell, and Moore 78]
J. McDermott, A. Newell, and J. Moore.
The Efficiency of Certain Production System Implementations.
In D. A. Waterman and F. Hayes-Roth (editors), *Pattern-Directed Inference Systems*, pages 155-176. Academic Press, New York, 1978.

[Michie 76] Donald Michie.
A Theory of Advice.
In E. W. Elcock and D. Michie (editors), *Machine Intelligence 8*, pages 151-168. Wiley, New York, 1976.

[Mitrinovic 64] D. S. Mitrinovic.
Elementrary Inequalities.
P. Noordhoff, Groningen, The Netherlands, 1964.

[Morgenstcin 80] Jacob P. Morgenstein.
Computer Based Management Information Systems Embodying Answer Accuracy as a User Parameter.
PhD thesis, University of California at Berkeley, December, 1980.

[Nickel 69] Karl Nickel.
Triplex-Algol and Its Applications.
In E. Hansen (editor), *Topics in Interval Analysis*, pages 10-24. Clarendon Press, Oxford, 1969.

[Nilsson 80] Nils Nilsson.
Principles of Artificial Intelligence.
Tioga, Palo Alto, 1980.

[Rall 81] L. B. Rall.
Interval Analysis: A Tool For Applied Mathematics.
Technical Report 2268, University of Wisconsin Mathematics Research Center, 1981.

[Rapopport 78] Anatol Rapopport.
Rank-Size Relations.
In W. Kruskal and J. Tanur (editors), *International Encyclopedia of Statistics*, pages 847-854. The Free Press, New York, 1978.

[Rips 75] Lance J. Rips.
Inductive Judgments about Natural Categories.
Journal of Verbal Learning and Verbal Behavior 14:665-681, 1975.

[Rosch and Mervis 75]
Eleanor Rosch and Carolyn B. Mervis.
Family Resemblances: Studies in the Internal Structure of Categories.
Cognitive Psychology 7:573-605, 1975.

[Rosch et al 76] Eleanor Rosch, Carolyn B. Mervis, Wayne D. Gray, David M. Johnson, and Penny Boyes-Braem.
Basic Objects in Natural Categories.
Cognitive Psychology 8:382-439, 1976.

[Rowe 81] Neil C. Rowe.
Rule-Based Statistical Calculations on a Database Abstract.
In *Proceedings*, pages 163-176. First LBL Workshop on Statistical Database Management, Menlo Park CA, December, 1981.

[Rowe 82] Neil C. Rowe.
Inheritance of Statistical Properties.
In *Proceedings of the National Conference*, pages 221-224. American Association for Artificial Intelligence, Pittsburgh PA, August, 1982.

[Rowe 83a] Neil C. Rowe.
Top-down Statistical Estimation on a Database.
In *Proceedings of the International Conference on Management of Data*, pages 135-145. ACM-SIGMOD, May, 1983.

[Rowe 83b] Neil C. Rowe.
Diophantine Compromise of a Statistical Database.
Information Processing Letters, 1983.
to appear.

[Shore and Johnson 81]
John E. Shore and Rodney W. Johnson.
Properties of Cross-Entropy Minimization.
IEEE Transactions on Information Theory IT-27(4):472-482, July, 1981.

[Shoshani 82] Arie Shoshani.
Statistical Databases: Characteristics, Problems, and Some Solutions.
In *Proceedings of the 8th Meeting*, pages 208-222. International Conference on Very Large Data Bases, 1982.

[Smith and Smith 77]
J. M. Smith and D. C. P. Smith.
Database Abstractions: Aggregation and Generalization.
ACM Transactions on Database Systems 2(2):105-133, June, 1977.

[Teitelman 75] Warren Teitelman et al.
Interlisp Reference Manual .
Xerox Palo Alto Research Center, Palo Alto, CA, 1975.

[Tobey 71] Robert C. Tobey.
Symbolic Mathematical Computation -- Introduction and Overview.
In S. R. Petrick (editor), *Proceedings of the Second Symposium*, pages 1-15. Symposium on Symbolic and Algebraic Manipulation, March, 1971.

[Tukey 77] John W. Tukey.
Exploratory Data Analysis.
Addison-Wesley, Reading, Mass., 1977.

[Ullman 80] J. D. Ullman.
Principles of Database Systems.
Computer Science Press, Potomac MD, 1980.

[Ullman 81] Jeffrey D. Ullman.
A View of Directions in Relational Database Theory.
Technical Report STAN-CS-81-852, Department of Computer Scieoce, Stanford University, May, 1981.

[vanMelle 80] William vanMelle.
A Domain-Independent System That Aids in Constructing Consultation Programs.
Technical Report STAN-CS-80-820, Stanford University Computer Science Department, 1980.

[Walker 80] Adrian Walker.
On Retrieval From a Small Version of a Large Data Base.
In *Proceedings of the 6th Meeting*, pages 47-54. International Conference on Very Large Data Bases, 1980.

[Wiederhold 77] Wiederhold, G.
Database Design.
McGraw-Hill, New York, 1977.

[Yu and Chin 77] C. T. Yu and F. Y. Chin.
A Study on the Protection of Statistical Databases.
In *Proceedings*, pages 169-181. ACM SIGMOD International Conference on Management of Data, 1977.

[Zadeh 79] L. Zadeh.
A Theory of Approximate Reasoning.
In E. W. Elcock and D. Michie (editors), *Machine Intelligence 9*, pages 149-196. Wiley, New York, 1979.